PARTNER AS TASK: BELIEF-CONDITIONED SUCCESSOR FEATURES FOR ZERO-SHOT COORDINATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Zero-shot coordination (ZSC) requires cooperation with partners unseen during training. We view each partner as a task: the reward remains fixed, but the partner changes the ego-agent's effective dynamics and hence how it should act. Learning these dynamics is difficult when the partner policy is unobserved, and reward is sparse. We introduce belief-conditioned successor features (BSFs), which summarize discounted future features conditioned on a belief over the partner's latent type. The belief summarizes the interaction history. We show that the value of a belief-independent policy is linear in the belief, which relates belief conditioning to task conditioning in standard successor features. We approximate BSFs in `MA-JEPA`, an on-policy self-play (SP) algorithm that combines a learned history encoder with a temporal difference latent prediction objective. This objective provides reward-free supervision for long-term predictive representations. Across Overcooked and SMAC, `MA-JEPA` outperforms SP, population-based, and agent-modeling baselines. Its largest cross-play gains over the strongest baseline in the main comparison appear under sparse rewards (77%) and RGB observations (32%).

Code: `https://anonymous.4open.science/r/MA-JEPA/`

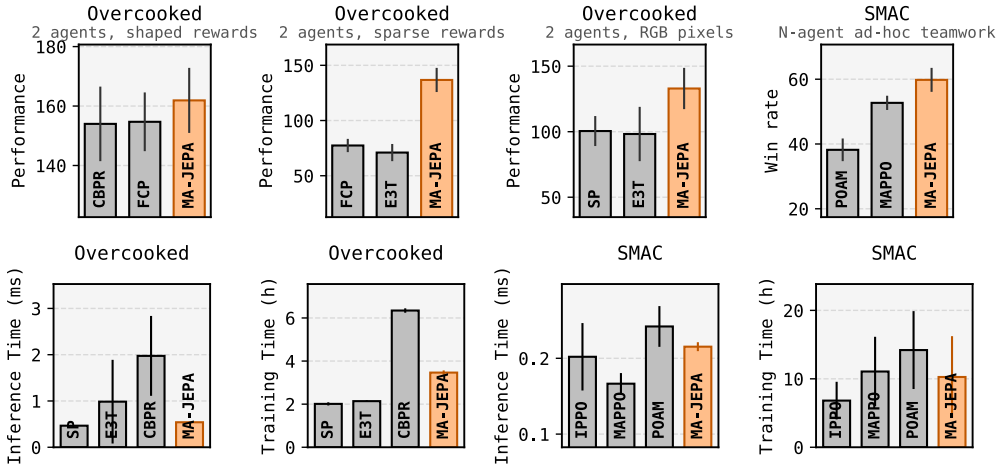

Figure 1: `MA-JEPA` improves ZSC performance across Overcooked and SMAC while adding modest computational overhead. (*Top*) Task performance in shaped-reward, sparse-reward, RGB-observation, and N-agent coordination settings. In this figure, we plot the top three performance methods for each configuration. (*Bottom*) Inference and training time, where lower is better.

1 INTRODUCTION

Zero-shot coordination (ZSC) in multi-agent reinforcement learning (MARL) studies cooperation with partners unseen during training. Common training schemes include self-play (SP) (Tesauro, 1994; Silver et al., 2017) and population-based training (Jaderberg et al., 2017). More recently, other approaches use theory-of-mind modeling (Rabinowitz et al., 2018), partner-action prediction (Yan et al., 2023), or cross-environment training (Jha et al., 2025). However, learning to coordinate remains

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

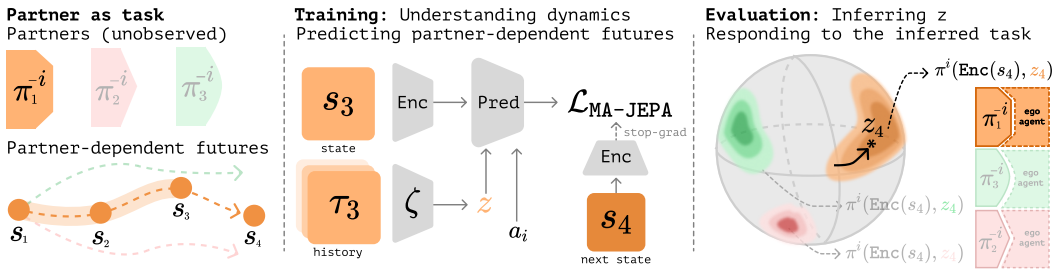

Figure 2: **Overview.** (*Problem*) The ego-agent interacts with a partner drawn from a partner pool, and the partner determines which future follows a given state. (*Training*) A history encoder ζ maps the trajectory to a belief embedding z, and the predictor is trained with a TD target to predict the discounted future under that partner, not the average. (*Evaluation*) The policy takes the state together with z, which summarizes the partner-dependent dynamics observed so far.

challenging under sparse rewards or high-dimensional observations. Sparse rewards provide limited feedback over long interaction sequences. Image observations also require agents to learn visual features alongside coordination behavior. These challenges motivate a research question: *what should an agent learn from interactions beyond the task reward?*

Our main hypothesis is that an agent should learn how its partner affects its own long-term outcomes. From the ego-agent's perspective, each partner induces a task, namely the MDP obtained by treating the partner as part of the environment. The reward remains fixed, but the effective transition dynamics, and thus how the ego-agent should act, depend on the partner. Since the partner policy is unobserved, predicting these outcomes requires information from the interaction history. Learning from transitions provides a way to capture these differences without reward supervision.

This motivates a connection to successor features (SFs) (Dayan, 1993; Barreto et al., 2017), which offer a representation of long-term outcomes as discounted feature accumulations. Standard SFs separate these predictions from the reward function to support transfer across tasks (Borsa et al., 2018). In ZSC, the situation is reversed: a change in partner changes the dynamics, not the reward. We address this by augmenting the state with a belief over the partner's latent type. Under our modeling assumptions, this belief summarizes the relevant history and yields a Markov process. We define belief-conditioned successor features (BSFs) on this process and show that the value of a belief-independent policy is linear in the belief, which relates belief conditioning to task conditioning.

We instantiate BSFs in `MA-JEPA`, an on-policy algorithm with SP. A history encoder turns the interaction history into a belief embedding, and a latent predictor is trained with a bootstrapped TD target instead of the single-step prediction used by JEPA (LeCun et al., 2022). Figure 2 summarizes the three stages. `MA-JEPA` changes what the ego-agent learns from each interaction, not which partners it encounters. In principle, it can be added on top of any exposure-based scheme, such as population-based training; in this work, we consider plain SP to isolate its effect.

Contributions. First, we formulate ZSC with partner-defined tasks and introduce BSFs on the belief-augmented state. Second, we theoretically analyze when the value function is linear in the belief and relate this decomposition to task conditioning in SFs. Third, we develop `MA-JEPA`, which outperforms the ZSC baselines of our main comparison on Overcooked and SMAC. Its gains over the strongest of them are about 77% under sparse rewards and 32% with RGB observations.

2 Related Works

Zero-shot coordination (ZSC). ZSC is typically evaluated via cross-play (XP) performance against unseen partners (Wang et al., 2024c). Most prior works have focused on expanding partner exposure during training. SP trains an ego-agent against itself (Tesauro, 1994; Silver et al., 2017; Hu et al., 2020; Yu et al., 2022; Lucas & Allen, 2022; Wang et al., 2023), thereby performing well in zero-sum settings, but can overfit to self-consistent conventions in cooperative games (Strouse et al., 2021). Next, PBT trains against a diverse pool of partners (Jaderberg et al., 2017; Vinyals et al., 2019; Carroll et al., 2019; Strouse et al., 2021; Sarkar et al., 2023), improving robustness at the cost of generating and maintaining the population. Recent advances incorporate theory of mind (Rabinowitz

et al., 2018; Ni et al., 2026), explicit prediction of partner actions (Yan et al., 2023; Lee & Kwon, 2024), meta-task inference (Wang et al., 2024b), and cross-environment training (Jha et al., 2025). These methods are complementary to ours; that is, they shape which partners the ego-agent trains against, while we shape what representation is learned from those interactions. In our experiments, `MA-JEPA` is trained with SP, yielding representations that support robust coordination across all these challenging conditions (*e.g.*, sparse reward and high-dimensional observation).

Partner modeling. Several methods infer the partner explicitly from the interaction history and condition the ego policy on this estimate. They learn policy representations (Grover et al., 2018) or theory-of-mind models (Rabinowitz et al., 2018), predict partner actions (Yan et al., 2023), or infer the partner's meta-task (Wang et al., 2024b). Meta-RL methods such as VariBAD (Zintgraf et al., 2021) and PEARL (Rakelly et al., 2019) infer a latent task in a similar way for single-agent task adaptation. These methods infer an identity, a strategy type, the next action, or a task parameter. Next-action targets look only one step ahead, and meta-RL encoders learn partly from rewards, which give little signal in sparse settings. Moreover, partners that act alike in the short term can still lead the ego-agent to different long-term outcomes. In contrast, the proposed solution models the partner implicitly. It learns a continuous history embedding only through its use in predicting the ego-agent's discounted future latents. This target needs no partner labels, fixed types, or rewards and reflects how the partner changes the ego-agent's long-horizon future, which is what the ego policy must adapt to.

Successor representation in RL. Our main idea in this work is based on the successor representation (Dayan, 1993) and its expansion, *including* SFs (Barreto et al., 2017) and successor measures (SMs) (Blier et al., 2021), in MARL. This provides a powerful framework for decoupling environment dynamics from task-specific reward functions (Agarwal et al., 2025; Sikchi et al., 2025). This decoupling has been used for transfer learning (Barreto et al., 2017; Borsa et al., 2018) and efficient exploration (Machado et al., 2018). Recent studies have begun extending SFs to multi-agent domains, *e.g.*, evaluating teammates under varying rewards (Yang et al., 2021), computing min-max values in competitive games (Amatya et al., 2025), and disentangling individual contributions (Kim et al., 2022). These works assume centralized training with access to all agents' actions or policies. In contrast, we focus on the ego-agent's perspective, where the partner's policy is unobserved, which introduces the non-Markovianity addressed by belief conditioning.

3 Preliminaries

Decentralized MDPs. A decentralized MDP (Bernstein et al., 2002) is a tuple $\langle \mathcal{I}, \mathcal{S}, \{\mathcal{A}_j\}_{j\in\mathcal{I}}, \mathcal{T}, r, \gamma\rangle$, where $\mathcal{I} = \{1, \dots, N\}$ is a set of agents, $\mathcal{S}$ the state space, $\mathcal{A}_j$ the action space of agent j, $\mathcal{T}(s' \mid s, a_1, \dots, a_N)$ the joint transition kernel, $r : \mathcal{S} \to \mathbb{R}$ a shared reward, and $\gamma \in [0, 1)$ the discount factor; each agent j acts according to its own policy π_j.

Hidden parameter MDPs (HiP-MDPs). A HiP-MDP (Doshi-Velez & Konidaris, 2016) is a single-agent MDP whose transition kernel $\mathcal{T}_\theta$ depends on a latent parameter $\theta \sim P_\Theta$ that is drawn once per episode and never observed. The agent must act well across the family $\{\mathcal{T}_\theta\}_{\theta\in\Theta}$ under a fixed reward.

Successor features (SFs). Consider θ to be fixed and the viewpoint of agent i. For any policy π_i and state-action pair (s, a_i),* we define the successor measure $M^\pi(\mathcal{U} \mid s, a)$, where $\mathcal{U} \subseteq \mathcal{S}$ is a subset of state space, as a representation of the discounted future state visitation distribution as follows:

$$M^\pi(U \mid s, a) = \sum_{t=0}^{\infty} \gamma^t \Pr\left(s_{t+1} \in U \mid s_0 = s, a_0 = a, \pi\right). \tag{1}$$

Given a reward function $r(s)$, this can be decomposed as a linear combination of state features $\psi : \mathcal{S} \to \mathbb{R}^{d_\psi}$ and a task-specific vector $w \in \mathbb{R}^{d_\psi}$, detailed in $r(s) = \psi(s)^\top w$. Based on this, the action-value function for a given reward function r can be expressed as follows:

$$Q_r^\pi(s, a) = \int_{\mathcal{S}} r(s')\, M^\pi(ds' \mid s, a). \tag{2}$$

While M^π fully captures the long-term dynamics, directly estimating this measure is often intractable. Alternatively, SFs $F^\pi(s, a)$ are defined as the integral of the features with respect to the successor

*For simplicity, we write individual notation as π and a in the rest of this section.

measure. By the linearity of the integral, Q can be written using $F^\pi(s,a)$ as follows:

$$Q^\pi_w(s,a) = \int_{\mathcal{S}} \psi(s')^\top w \, M^\pi(ds' \mid s,a) = \Big(\int_{\mathcal{S}} \psi(s') \, M^\pi(ds' \mid s,a) \Big)^\top w \equiv F^\pi(s,a)^\top w. \quad (3)$$

Latent predictive representation learning. In complex domains, it is useful to learn a compact state representation $\phi : \mathcal{S} \to \mathbb{R}^d$ that captures the underlying structure of the environment. The latent predictive approach posits that a good representation allows for the accurate prediction of future latent states (LeCun et al., 2022). Formally, a latent transition predictor $T : \mathbb{R}^d \times \mathcal{A} \to \mathbb{R}^d$ is trained jointly with ϕ to minimize the one-step prediction error in the latent space as follows:

$$\mathcal{L}_{\text{pred}}(\phi, T) = \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \mathcal{D}} \Big[\big\| T(\phi(s_t), a_t) - \text{sg}\left[\bar{\phi}(s_{t+1})\right] \big\|_2^2 \Big]. \quad (4)$$

where $\bar{\phi}$ is an EMA copy of ϕ and $\text{sg}[\cdot]$ denotes the stop-gradient operator; together, they are commonly used to avoid representation collapse (Schwarzer et al., 2021).

4 Problem Setup: Partner as Task

This section provides our view of the ZSC problem setup. Throughout, i denotes the ego-agent and $-i$ its partners, with $a_{-i} \in \mathcal{A}_{-i} := \prod_{j \neq i} \mathcal{A}_j$.

Decentralized HiP-MDP. We combine the two models of Section 3: *the partner is a hidden parameter of the ego-agent's dynamics*. Formally, we consider $\mathcal{M} = \langle \mathcal{I}, \mathcal{S}, \{\mathcal{A}_j\}_{j \in \mathcal{I}}, \mathcal{T}, r, \gamma, \Theta, P_\Theta \rangle$, where partner behavior is governed by a latent $\theta \in \Theta$ through a policy $\pi_{-i}(a_{-i} \mid s; \theta)$. Marginalizing over the partner's action gives the ego-agent's effective dynamics, as follows:

$$\mathcal{T}_\theta(s' \mid s, a_i) := \sum_{a_{-i} \in \mathcal{A}_{-i}} \mathcal{T}(s' \mid s, a_i, a_{-i}) \, \pi_{-i}(a_{-i} \mid s; \theta). \quad (5)$$

For a fixed θ, the ego-agent faces an ordinary MDP $\mathcal{M}_\theta = \langle \mathcal{S}, \mathcal{A}_i, \mathcal{T}_\theta, r, \gamma \rangle$, which we call the *task induced by partner* θ. This is the sense in which we treat the *partner as a task*. Unlike standard SFs, which share dynamics and differ in w, partner-induced tasks share the reward and differ only in dynamics, so the same (s, a_i) can lead to different futures under different partners. Following prior works (Zhang et al., 2021; Strouse et al., 2021), $\theta \sim P_\Theta$ is drawn once at the start of an episode and held fixed thereafter, and the partner acts according to the stationary policy in (5).

Zero-shot coordination. Training and evaluation draw partners from distributions P_{train} and P_{test}. Since θ is never observed, the ego policy may depend only on its own interaction history $\tau_t = (s_0, a_{i,0}, \ldots, s_t)$, *i.e.*, $\pi(a_{i,t} \mid \tau_t)$. Therefore, the objective can be written as follows:

$$J_{\text{test}}(\pi) = \mathbb{E}_{\theta \sim P_{\text{test}},\, \tau \sim (\mathcal{T}_\theta, \pi)} \Big[\sum_{t=0}^{\infty} \gamma^t r(s_{t+1}) \Big]. \quad (6)$$

At evaluation, network parameters are frozen, and no partner identity, partner policy, or reward label is provided. The agent may only update its internal summary of τ_t.

The challenge. Because θ is hidden, the process the ego-agent actually observes is non-Markovian.

Proposition 4.1 (Non-Markovianity of observed dynamics)**.** *For any ego policy,* $\Pr(s_{t+1} \mid \tau_t, a_{i,t}) = \int_\Theta \mathcal{T}_\theta(s_{t+1} \mid s_t, a_{i,t}) p(\theta \mid \tau_t) \mathrm{d}\theta$*, where* $p(\theta \mid \tau_t) \propto P_\Theta(\theta) \prod_{k<t} \mathcal{T}_\theta(s_{k+1} \mid s_k, a_{i,k})$*. Therefore, the one-step transition depends on* τ_t *through the posterior over* θ*, not only through* s_t*.*

The proof is in Appendix B. Two consequences follow: First, any predictor of s_{t+1} from $(s_t, a_{i,t})$ alone, *e.g.*, (4) averages over partners that the history could tell apart. Second, when rewards are sparse, the objective in (6) gives little signal about which parts of τ_t matter. We therefore want a representation that summarizes τ_t well enough to make prediction Markov. It should predict discounted future outcomes rather than a single step and be learnable from transitions without reward.

5 Belief-Conditioned Successor Features (BSFs)

In this section, we specify what the ego-agent should predict from its interactions. Our answer is to condition SFs on a belief over the partner. Section 5.1 defines belief-conditioned successor features (BSFs), Section 5.2 shows mathematical properties, and Section 5.3 provides a practical instantiation, `MA-JEPA`, of the proposed solution.

5.1 Main idea

For a fixed θ, SFs in $\mathcal{M}_\theta$ give the per-partner SFs $F^\pi_\theta(s, a_i)$ of a policy $\pi(a_i \mid s)$, and $Q^\pi_{\theta,w} = {F^\pi_\theta}^\top w$ when $r = \psi^\top w$. This is the long-horizon, reward-free prediction, but it is indexed by θ, which the ego-agent never observes. Our idea is to replace θ by what the ego-agent can compute from its history.

Definition 5.1 (Belief)**.** The belief at time t is the posterior $b_t := p(\theta \mid \tau_t) \in \Delta(\Theta)$. It starts from $b_0 = P_\Theta$ and is updated recursively by $b_{t+1}(\theta) \propto \mathcal{T}_\theta(s_{t+1} \mid s_t, a_{i,t})\, b_t(\theta)$.

The update does not involve the ego policy, whose factors cancel in the posterior because actions are chosen from τ_t without observing θ; actions affect the belief only through $\mathcal{T}_\theta$. We write $x_t := (s_t, b_t) \in \mathcal{X} := \mathcal{S} \times \Delta(\Theta)$ for the belief-augmented state and let the ego policy depend on it.

Definition 5.2 (Belief-conditioned successor features)**.** The BSFs of a policy π are defined as follows:

$$F^\pi(x, a_i) := \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t \psi(s_{t+1}) \,\Big|\, x_0 = x, a_{i,0} = a_i, \pi\Big] \in \mathbb{R}^{d_\psi}. \tag{7}$$

For $x = (s, b)$, the expectation is over $\theta \sim b$ and the interaction that $\mathcal{T}_\theta$ and π generate from s. The belief is updated as in Definition 5.1 along the way.

In other words, the BSF averages the unknown partner under the current belief and refines that belief as the interaction unfolds. It is long-horizon and reward-free by construction.

5.2 Theoretical properties

We state two properties of BSFs. The first shows that BSFs are well-defined and learnable.

Proposition 5.3 (BSFs are successor features of a belief MDP)**.** *For $x = (s, b)$, let $\bar{\mathcal{T}}(s' \mid x, a_i) := \int_\Theta \mathcal{T}_\theta(s' \mid s, a_i)\, b(\theta)\, d\theta$. Then, $\Pr(s_{t+1} \mid \tau_t, a_{i,t}) = \bar{\mathcal{T}}(s_{t+1} \mid x_t, a_{i,t})$ and x_{t+1} is a deterministic function of $(x_t, a_{i,t}, s_{t+1})$. Hence, $\{x_t\}$ is Markov and $\bar{\mathcal{M}} := \langle \mathcal{X}, \mathcal{A}_i, \bar{\mathcal{T}}, r, \gamma \rangle$ is an MDP. Moreover, F^π in (7) is the successor feature of π in $\bar{\mathcal{M}}$ as follows:*

$$F^\pi(x, a_i) = \mathbb{E}_{s' \sim \bar{\mathcal{T}}(\cdot \mid x, a_i),\, a'_i \sim \pi(\cdot \mid x')}\big[\psi(s') + \gamma F^\pi(x', a'_i)\big], \tag{8}$$

where $x' = (s', b')$. When $r(s) = \psi(s)^\top w$, we also have $Q^\pi_w(x, a_i) = F^\pi(x, a_i)^\top w$.

Intuitively, Proposition 5.3 states that conditioning on the belief undoes Proposition 4.1, as the didactic example in Appendix C illustrates. The ego-agent is back in an ordinary MDP on the belief-augmented state, and (8) can be estimated by bootstrapping from transitions alone, without reward labels. The second shows what conditioning on the belief buys.

Proposition 5.4 (Linearity in the belief)**.** *Let $\pi(a_i \mid s)$ be a policy that does not depend on the belief. For all (s, b, a_i), we have $F^\pi(s, b, a_i) = \int_\Theta F^\pi_\theta(s, a_i)\, b(\theta)\, d\theta$ and hence the following:*

$$Q^\pi_w(s, b, a_i) = \int_\Theta F^\pi_\theta(s, a_i)^\top w\; b(\theta)\, d\theta. \tag{9}$$

For a finite $\Theta = \{\theta_1, \ldots, \theta_K\}$, this reads $F^\pi(s, b, a_i) = \big[F^\pi_{\theta_1}(s, a_i) \cdots F^\pi_{\theta_K}(s, a_i)\big]\, b$.

Intuitively, Proposition 5.4 states that a BSF is a belief-weighted mixture of per-partner SFs: the value is linear in b exactly as it is linear in w in (3). Inferring the partner thus plays the role that specifying the reward weights plays in standard SFs, which motivates conditioning the predictor on the belief. For policies that adapt to b, the value is in general not linear in b, but the optimal such policy is at least as good as any belief-independent one, which is a special case. See Appendix B for all proofs.

5.3 Practical instantiation

There are two challenges with implementing a practical learning algorithm for BSFs. First, maintaining an explicit posterior b_t is intractable, as the Bayesian update in Definition 5.1 requires the likelihood $\mathcal{T}_\theta$ for every θ. Second, computing $F^\pi(x, a_i)$ requires evaluating $\bar{\mathcal{T}}$, *i.e.*, $\int_\Theta \mathcal{T}_\theta(\cdot \mid s, a_i) b_t(\theta) \mathrm{d}\theta$, at every step. We address both by optimizing the BSF objective in JEPA. Below, o_t denotes the ego-agent's observation (*i.e.*, $o_t = s_t$ under full observability) and $\tau_t = (o_0, a_{i,0}, \ldots, o_t)$ its history.

Algorithm 1 MA-JEPA Training

Initialize $\phi, \zeta, T_\phi, \pi, V$ and $\bar{\phi}, \bar{T}_\phi$
while not converged **do**
  $\mathcal{D} \leftarrow \textsc{Rollout}(\phi, \zeta, \pi, N, K)$ ▷ Algorithm 2
  # JEPA-STYLE REPRESENTATION
  Recompute $e_t, \zeta(\tau_t)$ with gradients
  Calculate $\mathcal{L}_{\text{rep}}$ via (10) and $\mathcal{L}_{\text{vc}}$ via (11)
  # UPDATE FULL NETWORKS
  Calculate actor and value loss via $\mathcal{L}_{\text{PPO}}$
  Update $\pi, V, \phi, \zeta, T_\phi$ to minimize (12)
  Update target network $\bar{\phi}, \bar{T}_\phi$ via EMA

Algorithm 2 On-policy Rollout

Require: ϕ, ζ, π, parallel envs N, horizon K
function ROLLOUT
  for $k = 0, \ldots, K-1$ **do**
    $e_k \leftarrow \phi(o_k)$
    $\zeta(\tau_k) \leftarrow \zeta(\tau_{k-1}, e_k, a_{i,k-1})$
    $a_{i,k} \sim \pi(e_k, \zeta(\tau_k))$
  return $\mathcal{D} = \{(o_k, a_{i,k}, o_{k+1}, r_k, \zeta(\tau_k))\}$

Learning a belief embedding. We parameterize the belief as a learned history encoder ζ that maps τ_t into a fixed-size *belief embedding* $z_t = \zeta(\tau_t) \in \mathbb{R}^{d_z}$. Rather than training ζ to approximate the posterior $p(\theta \mid \tau_t)$ directly, we train it end-to-end through the BSF prediction objective below. This objective encourages, but does not guarantee, the prediction sufficiency that Proposition 5.3 relies on, $\Pr(s_{t+1} \mid \tau_t, a_{i,t}) = \Pr(s_{t+1} \mid s_t, z_t, a_{i,t})$, at the level of expected discounted features. If z_t misses partner-relevant information, one prediction must serve histories generated by different partners, and the resulting residual error drives ζ toward a more informative summary.

BSF in JEPA. We address the second challenge by extending the one-step JEPA loss in (4) to a TD motivated by Bagatella et al. (2026). We condition the predictor T on z_t and replace the one-step target with a bootstrapped TD target, which captures the discounted future in Definition 5.2 as follows:

$$\mathcal{L}_{\text{rep}} = \mathbb{E}\Big[\big\|T(\phi(o_t), z_t, a_{i,t}) - \text{sg}\left[\bar{\phi}(o_{t+1}) + \gamma \bar{T}\big(\bar{\phi}(o_{t+1}), z_{t+1}, a_{i,t+1}\big)\right]\big\|_2^2\Big]. \tag{10}$$

where bars denote EMA target networks and $a_{i,t+1}$ is the next action of the current policy. If z_t were an exact belief and the target encoder were fixed, the fixed point would satisfy (8) with $\bar{\phi}$ in the role of ψ (Appendix B.2). Both are learned in practice, so T only approximates a BSF. For readability, (10) uses one encoder, while the implementation uses two heads with a symmetric loss (Appendix D.1). The encoder ζ is trained by both $\mathcal{L}_{\text{rep}}$ and the PPO loss below.

Collapse prevention. Since the predictor operates in the same space as its target, the encoder ϕ risks collapsing to a constant function. We mitigate this with (*i*) EMA target networks with stop-gradient (Morales-Brotons et al., 2024), (*ii*) the separate encoder heads above, and (*iii*) a variance-covariance regularizer on the encoder output (Bardes et al., 2022):

$$\mathcal{L}_{\text{vc}} = \frac{1}{d}\sum_{j=1}^{d} \max\left(0, 1 - \sqrt{\text{Var}[\phi(o)_j] + \epsilon}\right) + \frac{1}{d(d-1)}\sum_{j \neq k} \text{Cov}\big[\phi(o)_j, \phi(o)_k\big]^2. \tag{11}$$

where $\phi(o)_j$ denotes the j-th dimension of the encoder output, and statistics are taken over the batch. The first term encourages each feature dimension to maintain unit variance, while the second term decorrelates distinct dimensions. In practice, we use (*iii*) only for RGB observations ($\lambda = 0.1$) and set $\lambda = 0$ in the state-based settings.

Full objective. Both the policy $\pi(a_{i,t} \mid \phi(o_t), z_t)$ and the value function $V(\phi(o_t), z_t)$ take the belief embedding as input, so the policy can adapt to the inferred partner. We train end-to-end with clipped PPO (Schulman et al., 2017) and the JEPA loss:

$$\underbrace{\mathbb{E}\left[-\min\left(\rho_t \hat{A}_t, \text{clip}(\rho_t, 1 \pm \epsilon_{\text{clip}})\hat{A}_t\right) + \alpha_V\left(V(\phi(o_t), z_t) - \bar{V}_t\right)^2 - \alpha_{\text{ent}}\mathcal{H}[\pi]\right]}_{\mathcal{L}_{\text{PPO}}} + \underbrace{\alpha_{\text{rep}}\left(\mathcal{L}_{\text{rep}} + \lambda\mathcal{L}_{\text{vc}}\right)}_{\mathcal{L}_{\text{MA-JEPA}}}, \tag{12}$$

where $\rho_t = \frac{\pi(a_{i,t}|\phi(o_t), z_t)}{\pi_{\text{old}}(a_{i,t}|\phi(o_t), z_t)}$ is the importance ratio, $\hat{A}_t$ denotes the GAE advantage estimate (Schulman et al., 2016), $\bar{V}_t$ is the target value, and $\mathcal{H}$ is the policy entropy. MA-JEPA follows a SP-style training scheme (Tesauro, 1994), *i.e.*, both agents share all network parameters, while each maintains its own z_t from its own history. We describe the MA-JEPA training pipeline in Algorithm 1, and provide training details in Appendix D.1.

6 EXPERIMENTS

This section poses and answers several research questions. We first outline the testbeds, baselines, and evaluation methods. As the setting of the specific problem varies, we provide a detailed explanation corresponding to each question. All results are based on the average with a 95% confidence interval.

For **testbeds**, our main testbed is Overcooked (Carroll et al., 2019), which consists of tasks requiring a single recipe. The standard observations are structured as $9 \times 9 \times C$ tensors, where C encodes state features (*e.g.*, positions, orientations, entity statuses, and so on). Notably, we also scale the testbed to $\texttt{Overcooked}_{\text{RGB}}$ using $88 \times 88 \times 3$ RGB input. To our knowledge, this marks the first study of the ZSC on $\texttt{Overcooked}_{\text{RGB}}$ in this domain. Agents receive a goal reward upon the successful delivery. In addition to this sparse signal, subgoal rewards can be incorporated through reward shaping to facilitate more efficient learning, following prior works (Jha et al., 2025).

For **baselines**, we consider four ZSC algorithms, *including* SP (Carroll et al., 2019), FCP (Strouse et al., 2021), E3T (Yan et al., 2023), and CBPR (Wang et al., 2024b). All baselines are based on IPPO (De Witt et al., 2020). In particular, SP_{ego} trains an agent by continually pairing it with its own current policy.[†] FCP trains a robust best response against a diverse pool of historical SP_{pop} checkpoints, whereas its variant, FCP-T, only considers the final checkpoint. As the agent modeling branch, E3T infers the partner's action and CBPR infers the meta-task.

For **evaluation**, we consider three types of XP. First, XP_{self} measures ZSC performance across different trained seeds of each algorithm. Second, we measure XP_{SP} against 160 checkpoints of SP_{eval} (Wang et al., 2024c). Lastly, XP_{algo} evaluates inter-algorithm performance (*e.g.*, MA-JEPA $\times$ E3T) (Jha et al., 2025). We consider 8 random seeds for training and 100 episodes for evaluation.

6.1 IS MA-JEPA BETTER THAN SP, FCP, AND E3T ON ZSC IN A FIXED TASK?

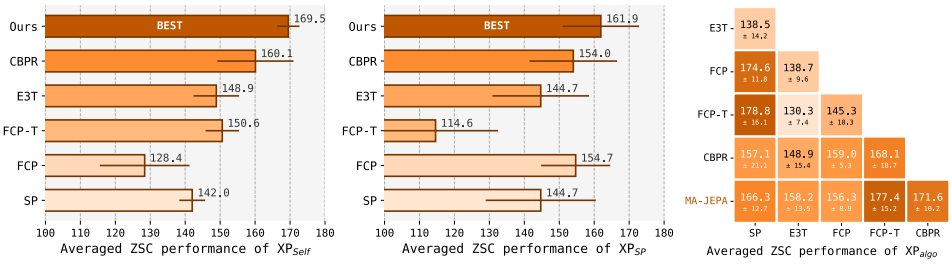

Figure 3: **Averaged XP performances over five layouts under a state-based fixed task setup.** (*Left*) ZSC performance of XP_{Self}. (*Center*) ZSC performance of XP_{SP}. (*Right*) ZSC performance of XP_{algo}. This shows bidirectional performance, so the upper and lower diagonals are the same.

Figure 3 reports the average XP scores across five Overcooked v1 layouts. MA-JEPA (ours) achieves the highest average XP score under all three evaluation protocols, although margins are moderate in this setting. In XP_{self}, MA-JEPA also exhibits the smallest variance across seeds, consistent with the auxiliary objective reducing sensitivity to initialization. This seed-level robustness is practically significant, as most prior works rely on a single best seed rather than reporting cross-seed consistency.

In XP_{SP}, MA-JEPA achieves 161.9, again the highest. Notably, FCP-T drops to 114.6 in this protocol despite its strong XP_{self} score. We attribute this to FCP-T training only against the final SP checkpoint, which leaves it poorly adapted to the lower-skill partners in the evaluation pool. In contrast, MA-JEPA shows only a small drop from XP_{self} to XP_{SP}, consistent with a representation that captures partner-dependent dynamics rather than a single convention.

In XP_{algo}, MA-JEPA achieves the highest average across all pairings. FCP and FCP-T show relatively strong coordination with SP-based agents, but do not generalize as well except for SP agents. MA-JEPA coordinates with agents from all algorithms, further supporting the hypothesis that structural environment understanding yields broadly compatible behavior.

[†]We distinguish three types of SP checkpoints: SP_{ego} for generating the ego agent, SP_{pop} for creating the checkpoint population for FCP training, and SP_{eval} for forming the SP pool for evaluation.

6.2 DOES MA-JEPA WORK EVEN IN A FULLY SPARSE SETUP WITHOUT REWARD SHAPING?

The objective of this question is to compare MA-JEPA against the baselines in a fully sparse, goal-conditioned environment without any intermediate subgoals.



Figure 4: **Learning curve and XP_{SP} under sparse reward setting without reward shaping.** Overcooked v1. (*Left*) Learning curve on episode returns. (*Right*) ZSC performance of XP_{SP}. The shaded area of the line plot and the cap of the bar plot denote 95% confidence interval.

Figure 4 presents the learning curves and XP_{SP} under the sparse reward setting. The learning curves reveal that MA-JEPA reaches roughly twice the return of the baselines in the Medium layout and is the only method with non-zero return in the Hard layout. This suggests that the SF-based prediction objective provides a useful learning signal even in the absence of shaped reward signals. The XP_{SP} results naturally show that MA-JEPA achieves the highest normalized score across all layouts.[‡]

Given the challenges of this setting, we also extend our analysis to compare MA-JEPA against several SP variations equipped with representation learning baselines (Laplacian (Wu et al., 2018), contrastive (Qiu et al., 2022), BYOL (Grill et al., 2020), and BYOL$-\gamma$ (Lawson et al., 2026)), which are widely used in RL domains. Despite sharing identical encoder and RL network structures, these representation objectives do not yield meaningful improvements over vanilla SP. Full results are provided in Appendix F.1.

6.3 HOW WELL DOES MA-JEPA MAINTAIN COORDINATION IN RGB IMAGE OBSERVATION?

We scale the observation space from symbolic state features to $88 \times 88 \times 3$ RGB pixels. All methods share an identical pixel encoder architecture to ensure a fair comparison.

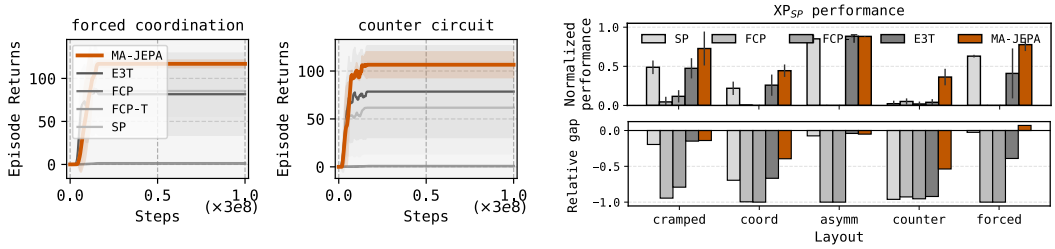

Figure 5: **Learning curve and XP_{SP} under RGB image observation.** $\text{Overcooked}_{\text{RGB}}$. (*Left*) Learning curve on episode returns. (*Right*, *Top*) ZSC performance of XP_{SP}. (*Right*, *Bottom*) The relative gap of XP_{SP} performance between a state-based input (section 6.1) and an image-based input. The shaded area of the line plot and the cap of the bar plot denote 95% confidence interval.

Figure 5 presents the XP_{SP} results under the RGB setting. Most strikingly, population-based methods (FCP and FCP-T) fail across all layouts. We conjecture that adapting to multiple partners during training is considerably harder with high-dimensional inputs. The encoder should learn to extract useful visual features and adapt to diverse partner behaviors, and without a structured representation learning signal, it rarely encounters enough successful coordination episodes to bootstrap learning.

Interestingly, SP achieves comparable performance, suggesting that the pixel encoder itself is not the bottleneck. When paired with a consistent partner, reward-driven learning can still extract sufficient features. However, MA-JEPA achieves the highest average score, demonstrating that the BSF-based objective is particularly valuable in the high-dimensional regime. Furthermore, MA-JEPA exhibits the smallest performance gap relative to its state-based counterpart (Section 6.1), indicating that the

[‡]The SP_{eval} pool used for this evaluation was trained with reward shaping, since SP trained under a fully sparse reward setup fails to learn a coherent role. Appendix E.2 explains this choice.

prediction objective effectively compensates for the increased observation complexity by providing a dense learning signal that guides the encoder toward task-relevant visual features.

6.4 How important is the belief encoder and BSF prediction for MA-JEPA?

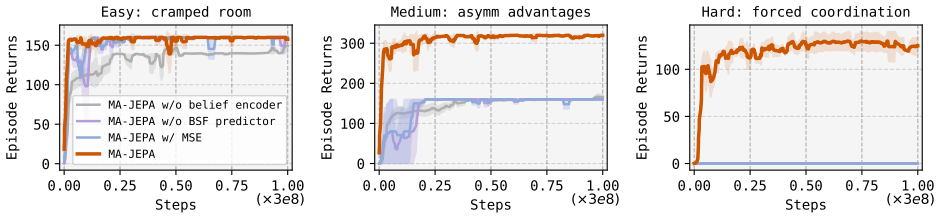

Figure 6: **Ablation study in the sparse setting.** We evaluate the individual contributions of the belief encoder and the predictor in MA-JEPA. MA-JEPA w/ MSE replaces TD target with a one-step target.

We study how each module contributes to MA-JEPA's learning efficiency, using training returns in the sparse setting (Figure 6). In the Easy layout, only removing the belief encoder lowers the final return. In the Medium and Hard layouts, removing either the belief encoder or the BSF predictor, or replacing the TD target with a one-step target (w/ MSE), reduces performance to the baseline level of Figure 4. Therefore, neither history conditioning nor one-step latent prediction suffices on its own; the gains require the belief encoder combined with the discounted TD target. Consequently, the full MA-JEPA achieves the highest and most stable returns across all difficulties.

6.5 Can MA-JEPA continue to function when we scale the number of partners?

Beyond two-player ZSC, we evaluate whether MA-JEPA extends to SMAC (Samvelyan et al., 2019) with up to 27 agents. In addition, this setting is partially observable. As a baseline, we consider POAM (Wang et al., 2024a), a recent N-agent ad-hoc teamwork algorithm. Table 1 reports the win rates on six maps, where each entry averages self-play and XP against 50% partners trained with the other algorithms. POAM performs competitively in small maps but degrades sharply as the number of agents increases. This suggests that explicitly modeling each partner becomes harder as joint behaviors multiply. On the other hand, MA-JEPA, which models partner influence implicitly, attains the highest or tied-highest mean win rate on all six maps, with clear margins on 5m_vs_6m and 10m_vs_11m, and is on par with MAPPO, which uses global information in training, on the largest map. Appendices F.9 and F.10 report further results and protocols.

Table 1: **N-agent ad-hoc teamwork on SMAC.**

Map	SP	MAPPO	POAM	MA-JEPA
3m	0.80 ± 0.04	0.92 ± 0.04	0.87 ± 0.04	0.92 ± 0.03
8m	0.67 ± 0.05	0.67 ± 0.05	0.40 ± 0.05	0.72 ± 0.05
2s3z	0.97 ± 0.02	0.97 ± 0.02	1.00 ± 0.02	1.00 ± 0.01
5m_vs_6m	0.05 ± 0.02	0.02 ± 0.02	0.02 ± 0.02	0.14 ± 0.04
10m_vs_11m	0.17 ± 0.03	0.12 ± 0.04	0.00 ± 0.00	0.34 ± 0.05
27m_vs_30m	0.15 ± 0.03	0.46 ± 0.04	0.00 ± 0.00	0.47 ± 0.04

7 Conclusion

We introduced the BSF, which extends successor features to multi-agent settings. This can restore the Markov property and decouple environment dynamics from task reward in a multi-agent setting. Our practical instantiation, MA-JEPA, combines PPO-based SP with a TD latent prediction loss and a learned history encoder. Across Overcooked and SMAC, MA-JEPA achieves the highest XP and self-play performance, with particularly large gains under sparse rewards (+77%) and RGB observations (+32%), where reward-driven representation learning alone is insufficient.

In this work, BSFs change the axis along which SFs generalize: we fix the reward and vary the partner through b. It remains open whether a single representation can vary both at once and let an agent transfer to a new partner and a new task together. Our analysis hints that this is possible because the value depends on the partner and on the reward separately, but we have shown it only for policies that ignore the belief and have not tested it. The analysis of Lehnert & Littman (2020), which generalizes across both transitions and rewards, may provide clues to this question. We hope the ideas in this work help address it and lead to agents that generalize over both partners and tasks.

AI USE STATEMENT

We used AI assistants to polish the writing and check grammar, and to check and refine our mathematical proofs. We also used them for engineering tasks, *including* writing job submission scripts, organizing the result tables and computing averaged scores, and drafting JAX implementations of baseline methods from their PyTorch implementations. The authors reviewed and verified all AI-assisted content and code, and take full responsibility for the paper.

ETHICS STATEMENT

This work presents foundational machine learning research and does not involve human subjects, sensitive personal data, or high-risk dual-use applications. The authors have evaluated the potential ethical implications of this work and anticipate no foreseeable negative societal impacts, safety risks, or ethical concerns beyond those standard to general computational research.

REPRODUCIBILITY STATEMENT

Anonymized code is available at the link in the abstract. Section 5.3 and Algorithm 1 describe the method; Appendices D.1∼D.2 give training, implementation, and architectural details; Appendix E describes the environments, observations, and rewards; and Appendix E.4 lists all hyperparameters.

REFERENCES

Siddhant Agarwal, Harshit Sikchi, Peter Stone, and Amy Zhang. Proto successor measure: Representing the behavior space of an RL agent. In *ICML*, 2025.

Sunny Amatya, Yi Ren, Zhe Xu, and Wenlong Zhang. Successor features for transfer in alternating Markov games. In *IROS*. IEEE, 2025.

Marco Bagatella, Matteo Pirotta, Ahmed Touati, Alessandro Lazaric, and Andrea Tirinzoni. TD-JEPA: Latent-predictive representations for zero-shot reinforcement learning. In *ICLR*, 2026.

Adrien Bardes, Jean Ponce, and Yann LeCun. Variance-invariance-covariance regularization for self-supervised learning. In *ICLR*, 2022.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P Van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *NeurIPS*, 2017.

Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Math. Oper. Res.*, 27(4):819–840, 2002.

Léonard Blier, Corentin Tallec, and Yann Ollivier. Learning successor states and goal-dependent values: A mathematical viewpoint. *arXiv preprint arXiv:2101.07123*, 2021.

Diana Borsa, André Barreto, John Quan, Daniel Mankowitz, Rémi Munos, Hado Van Hasselt, David Silver, and Tom Schaul. Universal successor features approximators. *arXiv preprint arXiv:1812.07626*, 2018.

Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-AI coordination. In *NeurIPS*, 2019.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Comput.*, 5(4):613–624, 1993.

Christian Schroeder De Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

Finale Doshi-Velez and George Konidaris. Hidden parameter Markov decision processes: A semiparametric regression approach for discovering latent task parametrizations. In *IJCAI*, 2016.

Michael O'Gordon Duff. *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes*. University of Massachusetts Amherst, 2002.

Jakob N Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *AAMAS*, 2018.

Tobias Gessler, Tin Dizdarevic, Ani Calinescu, Benjamin Ellis, Andrei Lupu, and Jakob Nicolaus Foerster. OvercookedV2: Rethinking overcooked for zero-shot coordination. *arXiv preprint arXiv:2503.17821*, 2025.

Dibya Ghosh, Chethan Anand Bhateja, and Sergey Levine. Reinforcement learning from passive data via latent intentions. In *ICML*, 2023.

Piotr J Gmytrasiewicz and Prashant Doshi. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*, 24:49–79, 2005.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. In *NeurIPS*, 2020.

Aditya Grover, Maruan Al-Shedivat, Jayesh Gupta, Yuri Burda, and Harrison Edwards. Learning policy representations in multiagent systems. In *ICML*, 2018.

Zhaohan Daniel Guo, Bernardo Avila Pires, Bilal Piot, Jean-Bastien Grill, Florent Altché, Rémi Munos, and Mohammad Gheshlaghi Azar. Bootstrap latent-predictive representations for multitask reinforcement learning. In *ICML*, 2020.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *ICML*, 2023.

Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. "Other-Play" for zero-shot coordination. In *ICML*, 2020.

Hengyuan Hu, Adam Lerer, Brandon Cui, Luis Pineda, Noam Brown, and Jakob Foerster. Off-belief learning. In *ICML*, 2021.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

Kunal Jha, Wilka Carvalho, Yancheng Liang, Simon S Du, Max Kleiman-Weiner, and Natasha Jaques. Cross-environment cooperation enables zero-shot multi-agent coordination. In *ICML*, 2025.

Seung Hyun Kim, Neale Van Stralen, Girish Chowdhary, and Huy T Tran. Disentangling successor features for coordination in multi-agent reinforcement learning. In *AAMAS*, 2022.

Daniel Lawson, Adriana Hugessen, Charlotte Cloutier, Glen Berseth, and Khimya Khetarpal. Self-predictive representations for combinatorial generalization in behavioral cloning. In *ICLR*, 2026.

Yann LeCun et al. A path towards autonomous machine intelligence version 0.9.2. *Open Review*, 62 (1):1–62, 2022.

Dongsu Lee and Minhae Kwon. Episodic future thinking mechanism for multi-agent reinforcement learning. *NeurIPS*, 2024.

Lucas Lehnert and Michael L. Littman. Successor features combine elements of model-free and model-based reinforcement learning, 2020. URL https://arxiv.org/abs/1901.11437.

Benjamin Li, Shuyang Shi, Lucia Romero, Huao Li, Yaqi Xie, Woojun Kim, Stefanos Nikolaidis, Charles Lewis, Katia Sycara, and Simon Stepputtis. Adaptively coordinating with novel partners via learned latent strategies. In *NeurIPS*, 2026.

Keane Lucas and Ross E Allen. Any-play: An intrinsic augmentation for zero-shot coordination. In *AAMAS*, 2022.

Long Ma, Yuanfei Wang, Fangwei Zhong, Song-Chun Zhu, and Yizhou Wang. Fast peer adaptation with context-aware exploration. In *ICML*, 2024.

Marlos C Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. Eigenoption discovery through the deep successor representation. In *ICLR*, 2018.

Daniel Morales-Brotons, Thijs Vogels, and Hadrien Hendrikx. Exponential moving average of weights in deep learning: Dynamics and benefits. *arXiv preprint arXiv:2411.18704*, 2024.

Andrew Ni, Simon Stepputtis, Stefanos Nikolaidis, Michael Lewis, Katia P Sycara, and Woojun Kim. Theory of mind guided strategy adaptation for zero-shot coordination. In *AAMAS*, 2026.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Shuang Qiu, Lingxiao Wang, Chenjia Bai, Zhuoran Yang, and Zhaoran Wang. Contrastive UCB: Provably efficient contrastive self-supervised learning in online reinforcement learning. In *ICML*, 2022.

Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, SM Ali Eslami, and Matthew Botvinick. Machine theory of mind. In *ICML*, 2018.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *ICML*, 2019.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *J. Mach. Learn. Res.*, 21(178):1–51, 2020.

Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Gardar Ingvarsson, Timon Willi, Ravi Hammond, Akbir Khan, Christian S de Witt, et al. JaxMARL: Multi-agent RL environments and algorithms in JAX. In *NeurIPS*, 2024.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

Bidipta Sarkar, Andy Shih, and Dorsa Sadigh. Diverse conventions for human-AI collaboration. In *NeurIPS*, 2023.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *ICLR*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *ICLR*, 2021.

Harshit Sikchi, Andrea Tirinzoni, Ahmed Touati, Yingchen Xu, Anssi Kanervisto, Scott Niekum, Amy Zhang, Alessandro Lazaric, and Matteo Pirotta. Fast adaptation with behavioral foundation models. In *RLC*, 2025.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. In *ICLR*, 2019.

Richard D Smallwood and Edward J Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations research*, 21(5):1071–1088, 1973.

DJ Strouse, Kevin McKee, Matt Botvinick, Edward Hughes, and Richard Everett. Collaborating with humans without human data. In *NeurIPS*, 2021.

Jayakumar Subramanian, Amit Sinha, Raihan Seraj, and Aditya Mahajan. Approximate information state for approximate planning and reinforcement learning in partially observed systems. *Journal of Machine Learning Research*, 23(12):1–83, 2022.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. In *AAMAS*, 2018.

Yunhao Tang, Zhaohan Daniel Guo, Pierre Harvey Richemond, Bernardo Avila Pires, Yash Chandak, Rémi Munos, Mark Rowland, Mohammad Gheshlaghi Azar, Charline Le Lan, Clare Lyle, et al. Understanding self-predictive learning for reinforcement learning. In *ICML*, 2023.

Gerald Tesauro. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Comput.*, 6(2):215–219, 1994.

Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. In *NeurIPS*, 2021.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Caroline Wang, Arrasy Rahman, Ishan Durugkar, Elad Liebman, and Peter Stone. N-agent ad hoc teamwork. *NeurIPS*, 2024a.

Haoming Wang, Zhaoming Tian, Yunpeng Song, Xiangliang Zhang, and Zhongmin Cai. Beyond single stationary policies: Meta-task players as naturally superior collaborators. In *NeurIPS*, 2024b.

Xihuai Wang, Zheng Tian, Ziyu Wan, Ying Wen, Jun Wang, and Weinan Zhang. Order matters: Agent-by-agent policy optimization. In *ICLR*, 2023.

Xihuai Wang, Shao Zhang, Wenhao Zhang, Wentao Dong, Jingxiao Chen, Ying Wen, and Weinan Zhang. ZSC-Eval: An evaluation toolkit and benchmark for multi-agent zero-shot coordination. In *NeurIPS*, 2024c.

Yifan Wu, George Tucker, and Ofir Nachum. The Laplacian in RL: Learning representations with efficient approximations. *arXiv preprint arXiv:1810.04586*, 2018.

Xue Yan, Jiaxian Guo, Xingzhou Lou, Jun Wang, Haifeng Zhang, and Yali Du. An efficient end-to-end training approach for zero-shot human-AI coordination. In *NeurIPS*, 2023.

Tianpei Yang, Weixun Wang, Hongyao Tang, Jianye Hao, Zhaopeng Meng, Hangyu Mao, Dong Li, Wulong Liu, Yingfeng Chen, Yujing Hu, et al. An efficient transfer learning framework for multiagent reinforcement learning. In *NeurIPS*, 2021.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. In *NeurIPS*, 2022.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander Smola. Deep sets. In *NeurIPS*, 2017.

Amy Zhang, Shagun Sodhani, Khimya Khetarpal, and Joelle Pineau. Learning robust state abstractions for hidden-parameter block MDPs. In *ICLR*, 2021.

Shuping Zhao, Bob Zhang, Jian Yang, Jianhang Zhou, and Yong Xu. Linear discriminant analysis. *Nature Reviews Methods Primers*, 4(1):70, 2024.

Luisa Zintgraf, Sebastian Schulze, Cong Lu, Leo Feng, Maximilian Igl, Kyriacos Shiarlis, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: Variational bayes-adaptive deep rl via meta-learning. *Journal of Machine Learning Research*, 22(289):1–39, 2021.

Appendix

CONTENTS

A Discussion **15**
A.1 Limitations and scope . . . 15
A.2 Connections to prior work . . . 15
A.3 Comparison with prior belief-based methods . . . 16

B Proofs **16**
B.1 Proof of Proposition 4.1 . . . 16
B.2 Proof of Proposition 5.3 . . . 17
B.3 Proof of Proposition 5.4 . . . 18

C Didactic example: `BSF` recovers Markov property under a decentralized HiP-MDP **18**

D Implementation details **18**
D.1 Training . . . 18
D.2 Architecture . . . 19

E Experimental details **19**
E.1 Environments . . . 19
E.2 Evaluation protocol . . . 20
E.3 Baselines . . . 21
E.4 Hyperparameters . . . 21

F Additional research questions **22**
F.1 Can generic representation objectives replace the BSF objective? . . . 22
F.2 How does `MA-JEPA` coordinate differently from `SP`? . . . 22
F.3 Does the belief encoder separate partners? . . . 22
F.4 How robust is `MA-JEPA` to a partner change within an episode? . . . 23
F.5 Can `MA-JEPA` be combined with a partner population? . . . 24
F.6 Does `MA-JEPA` coordinate across environments? . . . 24
F.7 Does `MA-JEPA` help with multiple recipes under partial observability? . . . 25
F.8 Does the `MA-JEPA` representation help adapt to new recipes? . . . 25
F.9 Does `MA-JEPA` work in general MARL settings? . . . 26
F.10 Does `MA-JEPA` work in N-agent ad hoc teamwork? . . . 26

A Discussion

A.1 Limitations and scope

Limitations. Our theory assumes an exact belief $b_t = p(\theta \mid \tau_t)$ but does not show that the learned encoder ζ recovers this posterior or a sufficient statistic of it. Namely, `MA-JEPA` is an approximate instantiation whose value we assess empirically. The ablation without the belief encoder, the partner separation in a linear projection of z_t, and the recovery after a partner switch all indicate that z_t keeps partner-dependent information. A promising way to close this gap is the theory of approximate information states, which bounds the loss in optimality when a history summary is only approximately sufficient (Subramanian et al., 2022). Extending such bounds to the BSF fixed point would tie the learned embedding to our guarantees. On the empirical side, a small task that admits an exact posterior would let a linear probe from z_t to b_t measure how much of the belief the encoder recovers over time. Proposition 5.4 also covers only belief-independent policies, so it motivates our design rather than guaranteeing the behavior of the trained agent.

Additionally, the implementation summarizes all teammates with one embedding. It remains competitive when SMAC teams mix teammates of different types but does not disentangle their individual types, and set-structured encoders over per-teammate histories (Zaheer et al., 2017) offer a natural way to factorize the belief.

Scope. We study adaptation within a single episode without extra interaction with the test partner. Following common practice in ZSC, the partner type stays fixed within an episode, so our Markov results do not cover partners that switch strategies. `MA-JEPA` still degrades gracefully under a mid-episode switch (Appendix F.4), and modeling the strategy as a latent state that can switch would extend BSFs to this case. Partners that keep learning in response to the ego-agent pose a further challenge that opponent-shaping methods (Foerster et al., 2018) begin to address. The evaluation pool is trained with reward shaping for all methods, even in the sparse setting (Appendix E.2). Finally, we build `MA-JEPA` on `SP` to isolate its effect, and its mixed results with a partner population (Appendix F.5) suggest that partner choice and representation learning deserve joint study.

A.2 Connections to prior work

Belief. A belief is classically a posterior over a hidden variable that makes the decision process Markov. In POMDPs, the hidden variable is the environment state (Smallwood & Sondik, 1973), and in Bayes-adaptive MDPs it is the unknown model (Duff, 2002). Our belief is closest to the latter but concerns the partner type θ, which stays fixed within an episode and reshapes the ego-agent's effective dynamics through the partner's policy. Proposition 5.3 applies the standard belief construction to this variable, and Proposition 5.4 mirrors the linearity of POMDP values in the belief for a fixed policy. Our addition is to condition successor features rather than values on the belief, which lets the belief play the role of the task vector and lets a reward-free objective shape it. `MA-JEPA` does not maintain an explicit posterior, since the likelihood $\mathcal{T}_\theta$ is unknown, and instead learns an embedding that is only encouraged to be sufficient for prediction. We also consider beliefs only over the partner type and not nested beliefs about what the partner believes, as in interactive POMDPs (Gmytrasiewicz & Doshi, 2005), or beliefs over hidden game information, as in off-belief learning (Hu et al., 2021).

Latent-predictive representations. Predicting future latent states is a common way to learn representations in RL without reconstructing observations. Self-predictive objectives train an encoder to predict its own future embeddings (Guo et al., 2020; Schwarzer et al., 2021), often against a slowly updated target network as in BYOL (Grill et al., 2020), and theory explains why such objectives avoid collapse (Tang et al., 2023). Discounted multi-step variants extend the prediction horizon (Lawson et al., 2026), and latent models also support planning and control (Hansen et al., 2023). JEPA (LeCun et al., 2022) frames this idea as prediction in a learned embedding space. A related line learns successor-style representations such as forward-backward representations (Touati & Ollivier, 2021) and intention-conditioned value functions (Ghosh et al., 2023). These methods assume a single agent with Markov dynamics, so their predictions average over partners when the partner is hidden, as Proposition 4.1 shows. The closest method is TD-JEPA (Bagatella et al., 2026), which combines JEPA-style prediction with a TD target to recover single-agent SFs. `MA-JEPA` follows this construction with three changes required by the partner. It augments the state with a belief, conditions the predictor on the interaction history, and collects data on-policy so that histories are not mixed across partners.

Contextualizing MA-JEPA in prior work. Existing ZSC methods mainly act on two levers. One lever is partner exposure, which diversifies training partners through self-play variants (Hu et al., 2020), populations (Strouse et al., 2021), or environments (Jha et al., 2025). The other lever is explicit partner modeling, which adds a module that predicts the partner's actions or type (Yan et al., 2023; Wang et al., 2024b). MA-JEPA acts on a third lever, namely what the ego-agent learns from each interaction, by training its representation to predict partner-dependent futures. This choice brings three practical benefits. It needs neither a partner population nor partner labels, so plain SP already yields strong cross-play. Its training signal does not depend on the reward, which matters most when rewards are sparse or observations are images. It also scales to many teammates without a separate model per partner, as the SMAC results in Section 6.5 suggest. Because these levers are complementary, combining them is a natural next step, although our combination with an FCP population in Appendix F.5 shows that the benefit depends on the layout.

A.3 Comparison with prior belief-based methods

Table 2 compares MA-JEPA with two recent methods that infer the partner explicitly. TALENTS (Li et al., 2026) tracks learned strategy types online and PACE (Ma et al., 2024) identifies the partner from multi-episode context, so we fit both to our single-episode protocol and report them apart from the main comparison. MA-JEPA is best in both settings, and its margin becomes large with sparse rewards. TALENTS is close to MA-JEPA with shaped rewards but falls far behind with sparse rewards, which suggests that identifying a partner from its actions is not enough when coordination depends on long-horizon outcomes. Online adaptation helps PACE in both settings but does not close this gap.

Table 2: **XP_{SP} of belief-based methods averaged over five layouts.**

Method	Shaped reward	Sparse reward
FCP	154.7	77.4
CBPR	154.0	49.4
TALENTS	160.3	72.8
PACE (w/o online adaptation)	123.9	68.1
PACE (w/ online adaptation)	153.8	80.8
MA-JEPA	161.9	136.7

B Proofs

Before deriving our mathematical properties, we assume the following. The ego-agent observes the state s_t. The partner type $\theta \sim P_\Theta$ is drawn once per episode, and given θ the partner acts according to the stationary policy $\pi_{-i}(\cdot \mid s; \theta)$ in (5). The initial state $s_0 \sim \rho_0$ is independent of θ. The policy of the ego agent is any $\pi(a_{i,t} \mid \tau_t)$ that does not observe θ. The feature map ψ is bounded. For discrete Θ, integrals over Θ are sums.

B.1 Proof of Proposition 4.1

Proof. We expand the one-step transition as follows:

$$\begin{aligned}\Pr(s_{t+1} \mid \tau_t, a_{i,t}) &= \int_\Theta \Pr(s_{t+1} \mid \tau_t, a_{i,t}, \theta) p(\theta \mid \tau_t, a_{i,t}) \mathrm{d}\theta \\ &= \int_\Theta \mathcal{T}_\theta(s_{t+1} \mid s_t, a_{i,t}) p(\theta \mid \tau_t) \mathrm{d}\theta, \end{aligned} \tag{13}$$

where the second equality uses that the process is Markov with kernel $\mathcal{T}_\theta$ given θ, and that $a_{i,t} \sim \pi(\cdot \mid \tau_t)$ is conditionally independent of θ given τ_t. Bayes' rule gives the posterior as follows:

$$\begin{aligned} p(\theta \mid \tau_t) \propto P_\Theta(\theta) p(\tau_t \mid \theta) &= P_\Theta(\theta) \rho_0(s_0) \prod_{k<t} \pi(a_{i,k} \mid \tau_k) \mathcal{T}_\theta(s_{k+1} \mid s_k, a_{i,k}) \\ &\propto P_\Theta(\theta) \prod_{k<t} \mathcal{T}_\theta(s_{k+1} \mid s_k, a_{i,k}), \end{aligned}$$

where the last step drops the factors that do not depend on θ. The right-hand side of (13) depends on τ_t through $p(\theta \mid \tau_t)$. Two histories with the same $(s_t, a_{i,t})$ but different posteriors give different values whenever $\mathcal{T}_\theta(\cdot \mid s_t, a_{i,t})$ varies with θ. Therefore, $\{s_t\}$ is not Markov in general. □

B.2 Proof of Proposition 5.3

Lemma B.1 (Trajectory law under the belief MDP). *Let $\bar{\mathcal{M}}$ be as in Proposition 5.3, $\pi(a_i \mid x)$ any policy on it, and $x_k = (s_k, b_k)$ with b_k given by Definition 5.1. For every $H \geq 1$ and $(x_0, a_{i,0}) = ((s, b), a_i)$, the trajectory law is as follows:*

$$\Pr(s_{1:H}, a_{i,1:H-1} \mid x_0, a_{i,0}) = \int_\Theta b(\theta) \Big[\prod_{k=0}^{H-1} \mathcal{T}_\theta(s_{k+1} \mid s_k, a_{i,k}) \prod_{k=1}^{H-1} \pi(a_{i,k} \mid x_k) \Big] \mathrm{d}\theta. \tag{14}$$

Proof. Let $Z_k := \int_\Theta b(\theta) \prod_{j<k} \mathcal{T}_\theta(s_{j+1} \mid s_j, a_{i,j}) \mathrm{d}\theta$, so that $Z_0 = 1$ and $b_k(\theta) = b(\theta) \prod_{j<k} \mathcal{T}_\theta(s_{j+1} \mid s_j, a_{i,j}) / Z_k$. The trajectory law expands as follows:

$$\begin{aligned}
\Pr(s_{1:H}, a_{i,1:H-1} \mid x_0, a_{i,0}) &= \prod_{k=0}^{H-1} \bar{\mathcal{T}}(s_{k+1} \mid x_k, a_{i,k}) \prod_{k=1}^{H-1} \pi(a_{i,k} \mid x_k) \\
&= \prod_{k=0}^{H-1} \frac{Z_{k+1}}{Z_k} \prod_{k=1}^{H-1} \pi(a_{i,k} \mid x_k) \\
&= Z_H \prod_{k=1}^{H-1} \pi(a_{i,k} \mid x_k) \\
&= \int_\Theta b(\theta) \Big[\prod_{k=0}^{H-1} \mathcal{T}_\theta(s_{k+1} \mid s_k, a_{i,k}) \prod_{k=1}^{H-1} \pi(a_{i,k} \mid x_k) \Big] \mathrm{d}\theta,
\end{aligned}$$

where the second equality uses $\bar{\mathcal{T}}(s_{k+1} \mid x_k, a_{i,k}) = \int_\Theta \mathcal{T}_\theta(s_{k+1} \mid s_k, a_{i,k}) b_k(\theta) \mathrm{d}\theta = Z_{k+1}/Z_k$, the third telescopes, and the last moves the policy factors inside the integral, which is valid because each x_k is a function of the history and not of θ. □

Proof of Proposition 5.3. The identity $\Pr(s_{t+1} \mid \tau_t, a_{i,t}) = \bar{\mathcal{T}}(s_{t+1} \mid x_t, a_{i,t})$ is (13) with $b_t = p(\theta \mid \tau_t)$, and $b_{t+1}(\theta) \propto \mathcal{T}_\theta(s_{t+1} \mid s_t, a_{i,t}) b_t(\theta)$ gives $x_{t+1} = (s_{t+1}, u(x_t, a_{i,t}, s_{t+1}))$ for a fixed map u. For any measurable $\mathcal{Y} \subseteq \mathcal{X}$, the conditional law of x_{t+1} is as follows:

$$\begin{aligned}
\Pr(x_{t+1} \in \mathcal{Y} \mid x_{0:t}, a_{i,0:t}) &= \mathbb{E}\big[\Pr(x_{t+1} \in \mathcal{Y} \mid \tau_t, a_{i,t}) \big| x_{0:t}, a_{i,0:t} \big] \\
&= \int_{\mathcal{S}} \mathbb{1}\big[(s', u(x_t, a_{i,t}, s')) \in \mathcal{Y}\big] \bar{\mathcal{T}}(\mathrm{d}s' \mid x_t, a_{i,t}),
\end{aligned}$$

where the first equality is the tower property, since $(x_{0:t}, a_{i,0:t})$ is a function of $(\tau_t, a_{i,t})$, and the second uses the two identities above. The result depends on the past only through $(x_t, a_{i,t})$, so $\{x_t\}$ is Markov. Since $r(s_{t+1})$ is a function of x_{t+1}, $\bar{\mathcal{M}}$ is an MDP.

The process in Definition 5.2 draws $\theta \sim b$ and then runs $\mathcal{T}_\theta$ and π with the belief updated along the way, so its trajectory law is the right-hand side of (14), which equals the law under $\bar{\mathcal{M}}$ by Lemma B.1. Hence F^π in (7) is the SF of π in $\bar{\mathcal{M}}$ and satisfies the following recursion:

$$\begin{aligned}
F^\pi(x, a_i) &= \mathbb{E}\Big[\psi(s_1) + \gamma \sum_{t=0}^{\infty} \gamma^t \psi(s_{t+2}) \Big| x_0 = x, a_{i,0} = a_i, \pi \Big] \\
&= \mathbb{E}_{s' \sim \bar{\mathcal{T}}(\cdot \mid x, a_i), a_i' \sim \pi(\cdot \mid x')} \big[\psi(s') + \gamma F^\pi(x', a_i') \big],
\end{aligned}$$

where the second equality uses the Markov property of $\{x_t\}$ and the tower property. Finally, $Q_w^\pi(x, a_i) = \mathbb{E}[\sum_t \gamma^t \psi(s_{t+1})^\top w \mid x, a_i, \pi] = F^\pi(x, a_i)^\top w$ by linearity of expectation. □

The recursion in (8) also determines F^π uniquely. Its right-hand side defines an operator on bounded functions of (x, a_i) that is a γ-contraction in the sup norm, since the ψ terms cancel and $\gamma < 1$. By the standard argument for Bellman operators (Puterman, 2014), F^π is therefore its unique bounded fixed point. This is the fixed point that the TD objective in (10) targets when z_t is an exact belief and the target encoder is fixed.

B.3 Proof of Proposition 5.4

Proof. We condition on the drawn θ in Definition 5.2 as follows:

$$F^{\pi}(s,b,a_i) = \int_{\Theta} \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t \psi(s_{t+1}) \Big| s_0 = s, a_{i,0} = a_i, \mathcal{T}_\theta, \pi\Big] b(\theta)\mathrm{d}\theta$$
$$= \int_{\Theta} F^{\pi}_{\theta}(s,a_i) b(\theta)\mathrm{d}\theta,$$

where the first equality uses Fubini's theorem, which applies since ψ is bounded and $\gamma < 1$, and the fact that $\pi(a_i \mid s)$ ignores the belief, so that the conditional interaction is that of π in $\mathcal{M}_\theta$. The second equality is the definition of the per-partner SF F^{π}_{θ} in $\mathcal{M}_\theta$. Then $Q^{\pi}_{w}(s,b,a_i) = F^{\pi}(s,b,a_i)^\top w = \int_\Theta F^{\pi}_{\theta}(s,a_i)^\top w b(\theta)\mathrm{d}\theta$, and for finite Θ the integral is the matrix-vector product $[F^{\pi}_{\theta_1}(s,a_i) \cdots F^{\pi}_{\theta_K}(s,a_i)]b$. □

Remark B.2 (Belief-conditioned policies). If π depends on b, conditioning on θ no longer yields the interaction of a fixed policy in $\mathcal{M}_\theta$: the actions depend on b_t, which is a function of the initial belief b and the history. Writing π_b for the resulting history-dependent policy, the same argument gives $F^{\pi}(s,b,a_i) = \int_\Theta F^{\pi_b}_{\theta}(s,a_i) b(\theta)\mathrm{d}\theta$, which is in general not linear in b because π_b itself depends on b. Nevertheless, every belief-independent policy is also a policy on $\bar{\mathcal{M}}$, so the optimal value over belief-conditioned policies is at least the optimal value over belief-independent ones.

C Didactic example: BSF recovers Markov property under a decentralized HiP-MDP

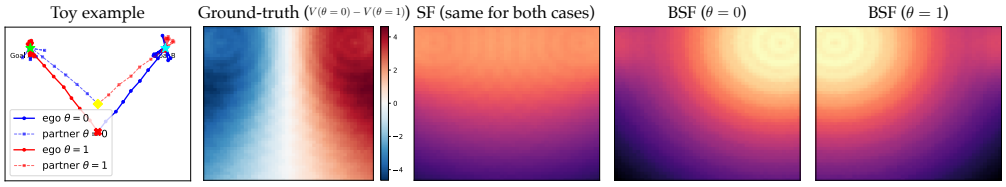

Figure 7: **A didactic example.** (*a*) sample trajectory on a two-agent landmark covering game at a given θ. (*b*) Ground-truth value difference $V(\theta = 0) - V(\theta = 1)$. (*c*) value map of standard SF. (*d, e*) value maps of the BSF at a given θ.

Figure 7 illustrates why the ego-agent needs a belief with a two-agent landmark covering task on a grid. The partner follows one of two strategies set by a hidden parameter θ, and the ego-agent must cover the landmark that the partner leaves free. The ground-truth values under the two partner types differ, so the optimal ego behavior depends on θ. Standard SFs average over both partner types and yield a single value map, which is the averaging that Proposition 4.1 predicts. BSFs conditioned on the belief instead recover a separate value map for each partner type, as Proposition 5.3 suggests.

D Implementation details

D.1 Training

Synchronous rollouts. We run N_{threads} parallel environments that each collect a fixed-length trajectory segment (Singh et al., 2019). All workers synchronize after each rollout and average their gradients into one update, so every rollout comes from the same policy as PPO requires. Averaging over independent trajectories also reduces the variance of the gradient estimate.

On-policy data. `MA-JEPA` trains on fresh rollouts rather than a replay buffer. In SP, the partner is a copy of the current policy, so the partner distribution changes during training. On-policy rollouts keep the TD targets consistent with this distribution and ensure that each history read by ζ comes from the current partner. Replayed histories would mix partner behaviors that no longer occur. PPO reuses each rollout for several epochs, so the data are strictly on-policy only for the rollout policy, and clipping keeps the updated policy close to it.

Dual encoder. Equation (10) uses one encoder and one predictor for readability. The implementation uses two heads ϕ and φ that share a CNN backbone but have separate MLP projections. The head $\phi(o_t) \in \mathbb{R}^{d_\phi}$ is the policy input, and $\varphi(o_t) \in \mathbb{R}^{d_\varphi}$ is the prediction target that plays the role of ψ in Section 5. Following TD-JEPA (Bagatella et al., 2026), a predictor T maps the ϕ-space to the φ-space, and a symmetric predictor $\tilde{T}$ maps the φ-space back to the ϕ-space. Both predictors also take the belief embedding and the action as inputs, and the implemented objective replaces (10) with the symmetric loss

$$\mathcal{L}_{\text{rep}} = \mathbb{E}\Big[\big\|T(\phi(o_t), z_t, a_{i,t}) - \text{sg}\big[\bar{\varphi}(o_{t+1}) + \gamma\, \bar{T}(\bar{\phi}(o_{t+1}), z_{t+1}, a_{i,t+1})\big]\big\|_2^2 + \big\|\tilde{T}(\varphi(o_t), z_t, a_{i,t}) - \text{sg}\big[\bar{\phi}(o_{t+1}) + \gamma\, \bar{\tilde{T}}(\bar{\varphi}(o_{t+1}), z_{t+1}, a_{i,t+1})\big]\big\|_2^2\Big], \tag{15}$$

where $\bar{\tilde{T}}$ is the EMA target of $\tilde{T}$. The first term is the BSF prediction in (10) with $\bar{\varphi}$ as the feature, and the second mirrors it with the roles of the two heads swapped. The two terms are weighted equally, and α_{rep} scales their sum. T and $\tilde{T}$ output d_φ and d_ϕ dimensions, respectively. The separate heads alone do not exclude a constant solution, so the EMA targets and, for RGB observations, the regularizer in (11) are also needed.

EMA targets. Each of ϕ, φ, T, and $\tilde{T}$ has an EMA target network updated as $\bar{\omega} \leftarrow (1-\eta)\bar{\omega} + \eta\omega$ with $\eta = 0.005$ after every gradient step. The TD targets in $\mathcal{L}_{\text{rep}}$ are computed only through these target networks with stop-gradient.

D.2 Architecture

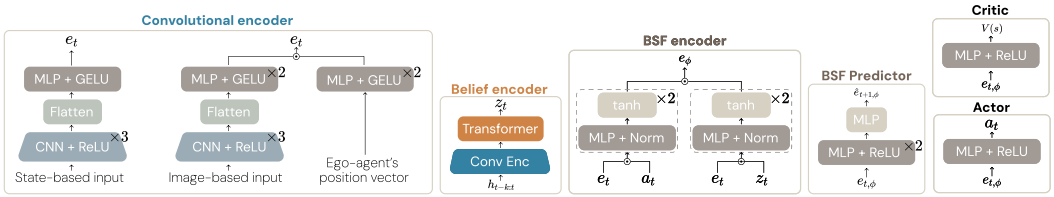

Figure 8: **Design of network architecture.**

Observation encoder. A shared CNN backbone feeds two linear heads for the ϕ-space ($d_\phi = 256$) and the φ-space ($d_\varphi = 64$). The state-based backbone has two convolutional layers with 2×2 kernels and stride 1, followed by a two-layer MLP with GELU. The RGB backbone uses the three-layer Nature DQN encoder followed by the same MLP. In the RGB setting, a 14-dimensional ego-position vector passes through a separate two-layer MLP and is concatenated with the CNN output. It only tells the policy which rendered agent it controls, since both agents appear in the same frame.

Belief encoder. The belief encoder ζ maps the history $\tau_t = (o_0, a_{i,0}, \ldots, o_t)$ to $z_t \in \mathbb{R}^{d_z}$, which stands in for b_t in Definition 5.1. It is a causal Transformer with N_{blk} blocks and context length L. Each input token concatenates $\phi(o_t)$ with a learned embedding of $a_{i,t-1}$ and is projected to dimension d_h with a learned positional embedding. The output of the last token is projected to d_z, and the token buffer is reset at episode boundaries.

BSF predictor. Each predictor has a state-action branch and a state-belief branch that a two-layer ReLU network merges. Both branches use LayerNorm followed by $\tanh$, which stabilized the TD targets better than ReLU in our experiments.

RL networks. The policy π and the value function V are MLPs on the concatenation of $\phi(o_t)$ and z_t. We use orthogonal initialization with scale $\sqrt{2}$ for hidden layers and 0.01 for the policy head.

E Experimental Details

E.1 Environments

Overcooked. We use Overcooked `v1` (Carroll et al., 2019; Rutherford et al., 2024) and `v2` (Gessler et al., 2025), a fully cooperative two-player cooking game with the layouts in Figure 9. Agents

Table 3: Network architecture of MA-JEPA.

Module	Layer
`Observation encoder (state)`	`Conv2d`$(64, k=2, s=1,$ `ReLU`$) \to$ `Conv2d`$(32, k=2, s=1,$ `ReLU`$)$ `Flatten` $\to$ `Dense`$(512,$ `GELU`$) \to$ `Dense`$(512,$ `GELU`$)$ ϕ-head: `Dense`(d_ϕ) / φ-head: `Dense`(d_φ)
`Observation encoder (RGB)`	`Conv2d`$(32, k=8, s=4,$ `ReLU`$) \to$ `Conv2d`$(64, k=4, s=2,$ `ReLU`$) \to$ `Conv2d`$(64, k=3, s=1,$ `ReLU`$)$ `Flatten` $\to$ `Dense`$(512,$ `GELU`$) \to$ `Dense`$(512,$ `GELU`$)$
	Position vector input (14-dim) $\to$ `Dense`$(64,$ `ReLU`$) \to$ `Dense`$(64,$ `ReLU`$)$
	`Concat` $\to$ ϕ-head: `Dense`(d_ϕ) / φ-head: `Dense`(d_φ)
`Belief encoder`	`Concat`$(\mathbf{e}_\phi,$ `Embed`$(a_{t-1})) \to$ `Dense`$(d_h,$ `ReLU`$)$ + `PosEmb` $[$`LayerNorm` $\to$ `Attn` $\to$ `LayerNorm` $\to$ `FFN`$(4d_h,$ `GELU`$)] \times N$ `LayerNorm` $\to$ `Dense`(d_z)
`BSF predictor`	SA branch: `Concat`$(\mathbf{e}, \mathbf{a}) \to [$`Dense`$(256) \to$ `LayerNorm` $\to \tanh] \times 2$ SB branch: `Concat`$(\mathbf{e}, \mathbf{z}_t) \to [$`Dense`$(256) \to$ `LayerNorm` $\to \tanh] \times 2$ Merge: `Concat` $\to$ `Dense`$(512,$ `ReLU`$) \to$ `Dense`$(512,$ `ReLU`$) \to$ `Dense`(d_{out})
`RL network`	Actor: `Dense`$(256) \to$ `Dense`$(192) \to$ `Dense`$(128) \to$ `Dense`$(64) \to$ `Dense`$(\lvert\mathcal{A}\rvert)$ (`ReLU`) Critic: `Dense`$(512) \to$ `Dense`$(256) \to$ `Dense`$(192) \to$ `Dense`$(128) \to$ `Dense`(1) (`ReLU`)

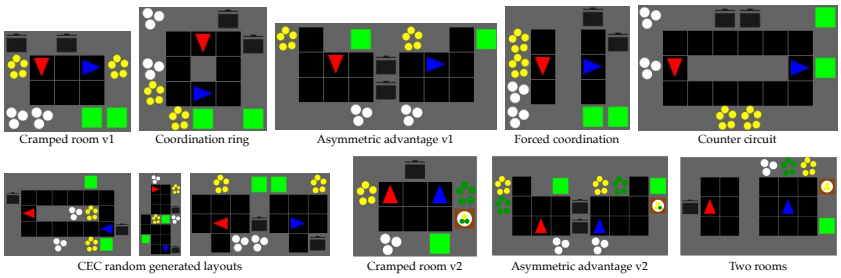

Figure 9: **Overcooked environment rendering.**

cook and deliver dishes in a shared kitchen, which requires implicit role assignment and avoiding each other. Overcooked `v1` has five layouts of increasing difficulty. Overcooked `v2` adds multiple ingredients, a recipe indicator that agents can query at a cost, and new layouts such as `Two rooms`. The CEC setting uses randomly generated layouts from Jha et al. (2025).

Observation. Table 4 details the state observations, which are $H \times W \times C$ grids with $C = 26$ channels in `v1` and $C \geq 30$ in `v2`. The extra channels in `v2` encode inventories and item states, and the grid is 9×9 in `v1`. In the partially observable `v2` experiments of Appendix F.7, each agent observes a $5 \times 5 \times C$ window centered on itself.

Reward. Agents receive a reward for each delivery, and Table 5 lists the intermediate rewards that reward shaping adds for subtasks such as placing ingredients and picking up plates. In `v2`, pressing the recipe indicator gives a penalty of -5. The sparse setting disables all intermediate rewards.

E.2 Evaluation protocol

Evaluation pool. The $\texttt{SP}_{\text{eval}}$ pool contains 160 checkpoints of `SP` agents from different seeds, taken at random initialization and at early, middle, and final stages of training (Carroll et al., 2019; Strouse et al., 2021). It thus spans a range of skill levels and conventions. No ego agent trains with this pool or with agents of other algorithms.

Table 4: Input structure of Overcooked v1 and v2. States are encoded as $9 \times 9 \times C$ grids.

Channel group	Content	v1	v2
Agent positions	Self / other location	2	2
Agent orientations	4-dir one-hot $\times$ 2 agents	8	8
Agent inventory	Held item encoding (plate, cooked, ingredient)	-	8
Static objects	Wall, pot, counter, goal, piles	6	6
Ingredient piles	Onion, tomato, plate source	3	1+
Pot state	Item counts, cook timer, done flag	6	1
Dynamic items	Items on counters (plate, onion, tomato)	-	4
Urgency	Remaining steps ≤ 40	1	-
Total C		26	30+

Table 5: Reward structure of Overcooked v1 and v2. Shaped rewards are used in the default setting; the sparse setting uses only the delivery reward.

Event	v1	v2	Sparse
Delivery (goal completion)	$+20$	$+20$	$+20$
Ingredient placement in pot	$+10$	$+3$	-
Plate pickup	$+3$	$+3$	-
Soup / dish pickup	$+15$	$+5$	-
Pot start cooking	-	$+5$	-
Recipe indicator button	-	-5	-

Reward shaping of the pool. In several layouts, both agents must take complementary roles before any reward appears. SP agents trained with sparse rewards alone often never learn such a role, and an evaluation against them would measure the failure of the partner rather than the adaptation of the ego agent. We therefore train the pool with reward shaping in every setting and never use shaping at evaluation. All methods thus face the same frozen partners with the same metric, and no ego agent is trained with shaping in the sparse setting.

Confidence intervals. For each of the 8 seeds, we average the score over evaluation partners and 100 episodes. We report the mean scores with the 95% confidence interval $\bar{x} \pm 2.36\, s/\sqrt{8}$, where s is their sample standard deviation. Figure 10 instead shows two standard deviations over seeds.

E.3 Baselines

We use JaxMARL (Rutherford et al., 2024) for Overcooked v1 and its SMAX reimplementation for SMAC. Our Overcooked v2 implementation follows https://github.com/FLAIROx/JaxMARL. The baselines follow their official implementations below.

- SP (IPPO) and FCP: https://github.com/sjtu-marl/ZSC-Eval
- MAPPO: https://github.com/zoeyuchao/mappo
- E3T: https://github.com/yanxue7/E3T-Overcooked
- CBPR: https://github.com/AlexWanghaoming/CBPR
- CEC: https://github.com/KJha02/crossEnvCooperation
- IQL (DRQN): https://github.com/keep9oing/DRQN-Pytorch-CartPole-v1
- VDN and QMIX: https://github.com/oxwhirl/wqmix
- POAM: https://github.com/carolinewang01/naht

E.4 Hyperparameters

Table 6 lists the training hyperparameters. Environment parameters such as the number of parallel environments, the rollout horizon, and the total timesteps follow prior work (Wang et al., 2024c; Rutherford et al., 2024; Jha et al., 2025; Gessler et al., 2025). PPO hyperparameters follow the defaults of JaxMARL (Rutherford et al., 2024) and MAPPO (Yu et al., 2022).

Table 6: Training hyperparameters for state, RGB, sparse, and CEC setups under Overcooked v1 and for multi-task setup under Overcooked v2.

Hyper-parameter	State	RGB	Sparse	CEC	Multi
Total timesteps	3.0×10^7	3.0×10^8	3.0×10^8	1.0×10^9	3.0×10^8
# of environments	256	64	256	256	256
# of steps	256	256	256	256	256
Epochs	4	4	4	4	4
# of minibatch	8	8	8	8	8
Learning rate	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}
Optimizer	Adam	Adam	Adam	Adam	Adam
Reward shaping	True	True	False	True	True
γ	0.95	0.95	0.95	0.95	0.95
GAE λ	0.95	0.95	0.95	0.95	0.95
ϵ clipping	0.2	0.2	0.2	0.2	0.2
α_{v}	0.5	0.5	0.5	0.5	0.5
α_{ent}	0.05	0.05	0.05	0.05	0.05
ϕ dim	256	256	256	256	256
φ dim	64	64	64	64	64
z dim	64	64	64	64	64
η (EMA)	0.005	0.005	0.005	0.005	0.005
α_{rep}	0.25	0.25	0.25	0.25	0.25
λ	0.0	0.1	0.0	0.0	0.0

F Additional research questions

F.1 Can generic representation objectives replace the BSF objective?

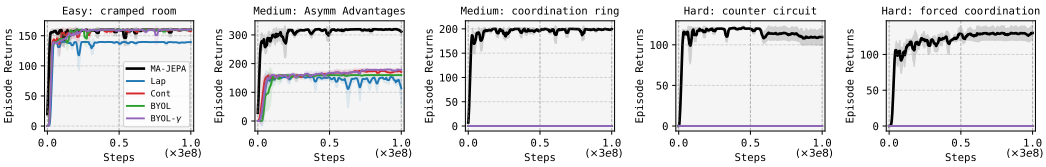

Figure 10: **Comparison against representation learning baselines under sparse rewards.** We plot the average learning curve, and the shaded area denotes two std over eight random seeds.

Figure 10 compares MA-JEPA under sparse rewards with SP agents trained with the Laplacian (Wu et al., 2018), contrastive (Qiu et al., 2022), BYOL (Grill et al., 2020), and BYOL-γ (Lawson et al., 2026) objectives. All variants share the encoder and RL networks of MA-JEPA. None of them approaches MA-JEPA beyond the Easy layout, as they fall well short of it in Asymmetric advantages and obtain no return in the remaining layouts.

F.2 How does MA-JEPA coordinate differently from SP?

Figures 11 to 13 compare the behavior of MA-JEPA and SP under sparse rewards through trajectories, state visitation, and value maps. We focus on the Medium and Hard layouts, where only MA-JEPA obtains substantial returns in Section 6.2. MA-JEPA agents take distinct roles and move along complementary paths, while SP agents often block each other or loop in Coordination ring and Counter circuit. The visitation maps show the same division of labor, and MA-JEPA agents also learn complementary value maps while SP agents assign high values to nearly all positions.

F.3 Does the belief encoder separate partners?

Figure 14 tests whether the belief embedding z_t separates partners by projecting it to two dimensions with linear discriminant analysis (Zhao et al., 2024). We roll out a MA-JEPA ego-agent with diverse partners and project z_t. Partner clusters overlap before any interaction and separate into distinct

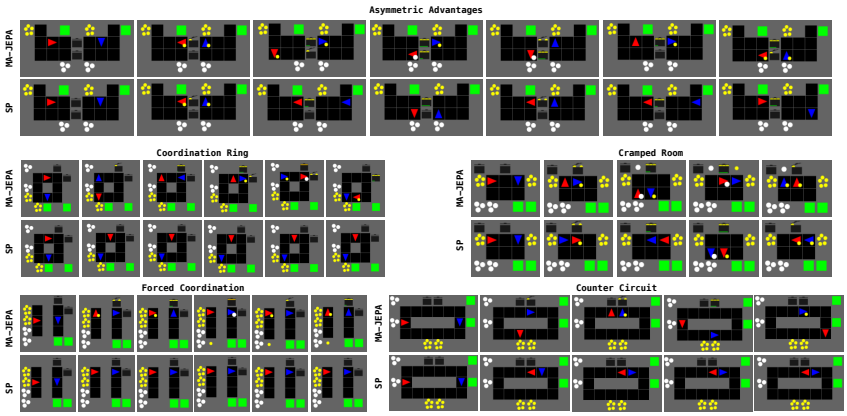

Figure 11: **Trajectory comparison.** We visualize trajectories according to layouts and algorithms. Each transition sample is selected from the entire trajectory periodically, *e.g.*, once every 10 timesteps.

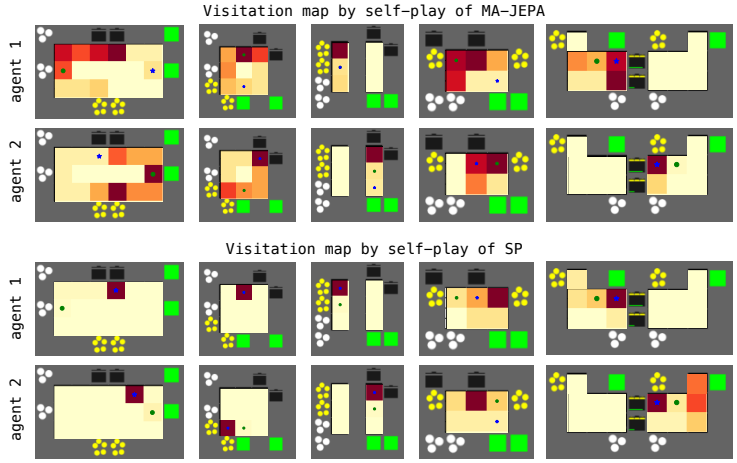

Figure 12: **Visitation occupancy map.** We plot the visitation occupancy of each agent, trained by a specific algorithm, as a heatmap on the overcooked map. The darker the color, the more frequent the visits; conversely, the lighter the color, the fewer the visits.

groups as the interaction proceeds in every layout. Some partners stay close, likely because they behave similarly in that layout. Since LDA uses partner labels and ζ is also trained by the PPO loss, this view is optimistic and does not isolate the contribution of the prediction objective.

F.4 How robust is MA-JEPA to a partner change within an episode?

Table 7: **XP_{SP} when the partner is replaced in the middle of an episode.**

Method	No switch	Early ↔ Late	Same maturity
SP	144.7	100.9	140.5
FCP	154.7	121.7	155.0
MA-JEPA	161.9	143.6	163.3

Table 7 studies how MA-JEPA copes when the partner is replaced in the middle of an episode. The history of the ego agent is not reset, and we average both switch directions between checkpoints of different or similar maturity in the state-based setting. After a switch between early and late checkpoints, MA-JEPA loses less than the reward of a single delivery, while SP and FCP lose considerably more. A switch between partners of similar maturity barely changes the score of any method.

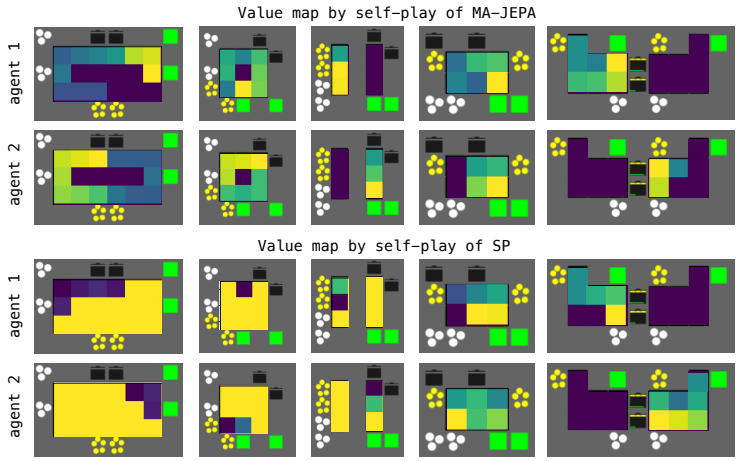

Figure 13: **Value map.** We plot the value map of each agent as a heatmap on the overcooked map. The darker the color, the lower the state value; the more yellow the color, the higher the state value.

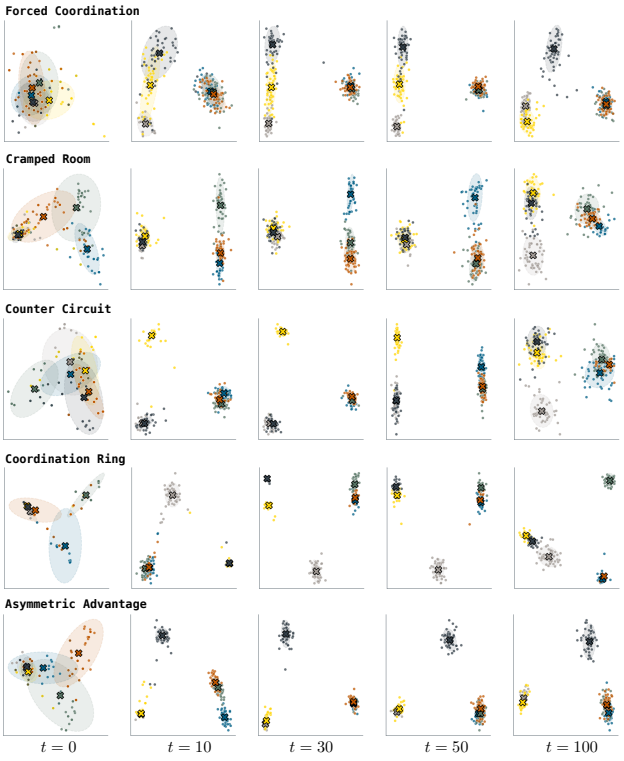

Figure 14: **Partner separation in the learned belief space.**

F.5 Can MA-JEPA be combined with a partner population?

Table 8 compares MA-JEPA trained with SP and with an FCP partner population under sparse rewards. The population raises the average only slightly, and its effect depends on the layout. It helps in Coordination ring, Asymmetric advantages, and Counter circuit but hurts in Cramped room and Forced coordination. MA-JEPA is thus complementary to population-based training rather than tied to one way of generating partners.

F.6 Does MA-JEPA coordinate across environments?

Figure 15 evaluates coordination across environments under the CEC protocol (Jha et al., 2025), which tests agents on randomly generated layouts. MA-JEPA has the fewest collisions in training, self-

Table 8: **XP_{SP} of MA-JEPA trained with SP or with an FCP population under sparse rewards.** Values are means and 95% confidence intervals.

Method	Cramped	Coordination	Asymmetric	Counter	Forced	Average
MA-JEPA (SP)	135.3±8.3	146.6±13.5	223.8±20.5	92.0±4.1	86.0±8.3	136.74
MA-JEPA (FCP)	120.5±12.0	173.8±5.5	243.9±10.8	110.4±10.6	48.8±15.5	139.48

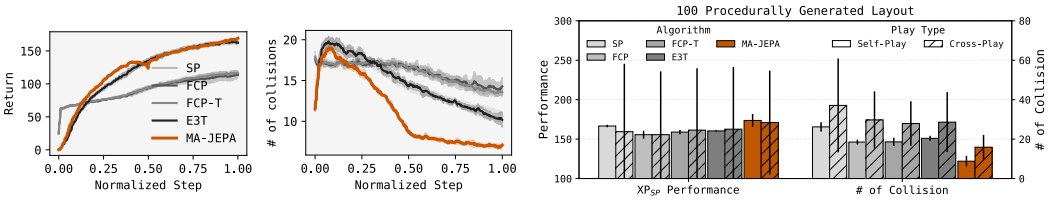

Figure 15: **Cross-environment cooperation performance.** We visualize performance in terms of return and the number of collisions.(*Left*) Learning curves. (*Right*) XP and self-play performance.

play, and cross-play while reaching returns similar to SP and E3T. FCP and FCP-T learn faster early but plateau lower, likely because their populations come from a fixed set of layouts. We conjecture that predicting long-term occupancy requires modeling where the partner will move, which helps the agent avoid spatial conflicts.

F.7 Does MA-JEPA help with multiple recipes under partial observability?

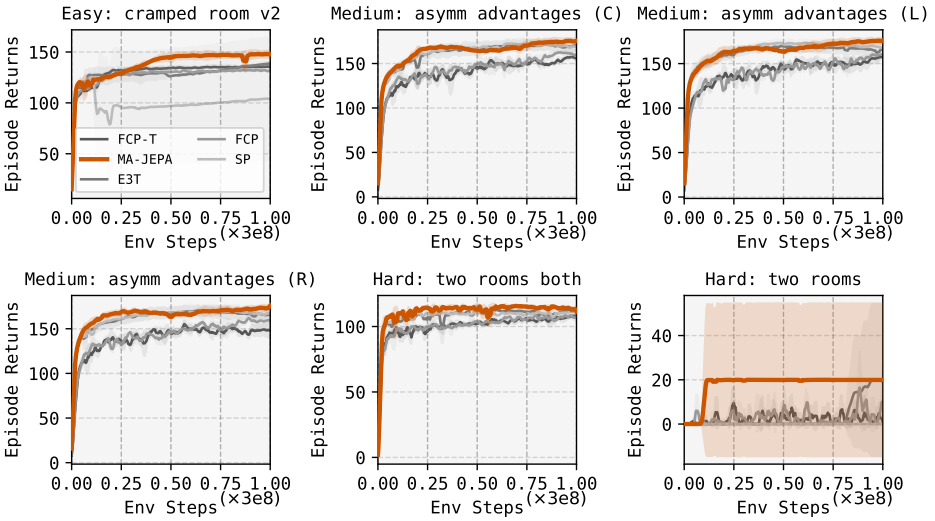

Figure 16: **Learning curves on Overcooked v2.**

Figure 16 reports self-play learning curves on Overcooked v2 with multiple recipes and partial observability. Each agent sees only a small window around itself, and the task here refers to the recipe rather than the partner-induced task of Section 4. MA-JEPA reaches higher or comparable returns in all configurations, with the clearest gap in Cramped room v2. Aggregating the local history in ζ may partly compensate for the missing global context.

F.8 Does the MA-JEPA representation help adapt to new recipes?

Figure 17 studies whether the MA-JEPA representation helps adaptation when a new recipe is introduced at each training phase on Overcooked v2. All methods drop at each phase boundary, and

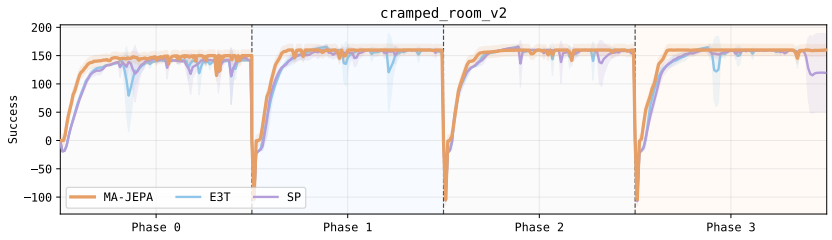

Figure 17: **Multi-task adaptation.** We plot learning curves about three algorithms trained on `cramped room v2` of Overcooked `v2`. Each phase means each recipe (*i.e.*, task) for training.

`MA-JEPA` recovers faster than `SP` and at least as fast as `E3T`. We hypothesize that the partner and dynamics structure learned by the BSF objective is largely reusable across recipes, so that mainly the reward-specific parts must be relearned.

F.9 Does MA-JEPA work in general MARL settings?

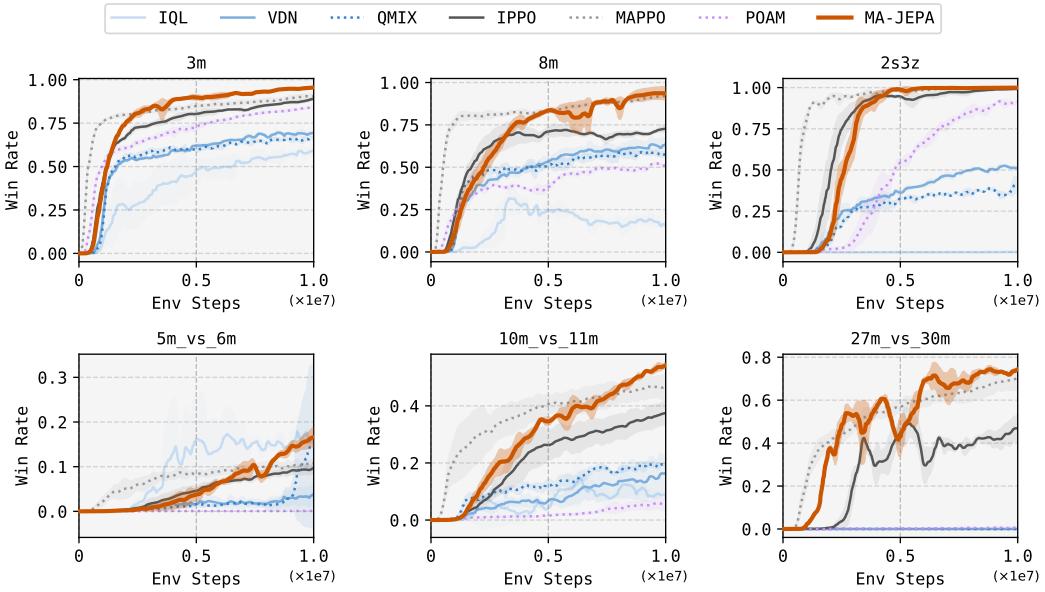

Figure 18: **Learning curves on SMAC.** The algorithms marked as a dotted line use global information for their training. The algorithms colored in blue and gray denote Q-family and PPO-family. We color the N-agent ad-hoc teamwork algorithm in purple.

Figure 18 compares `MA-JEPA` with `IPPO`, `MAPPO`, `POAM`, and value-based methods on SMAC maps of increasing team size (Samvelyan et al., 2019; Rutherford et al., 2024). The value-based methods are `IQL`, `VDN` (Sunehag et al., 2018), and `QMIX` (Rashid et al., 2020). `MA-JEPA` achieves the highest or comparable win rates on all maps, and its advantage over the decentralized `IPPO` grows with the team size. `MAPPO`, which uses global information in training, remains competitive, while `POAM` and the value-decomposition methods degrade on the larger maps.

F.10 Does MA-JEPA work in N-agent ad hoc teamwork?

Table 9 and Figure 19 report N-agent ad hoc teamwork on SMAC, where each algorithm is paired with every algorithm including itself. Unlike Table 1, which averages self-play and XP with half-teams from other algorithms, this table averages over all partner algorithms. `MA-JEPA` has the highest average on most maps, although its confidence interval overlaps with that of `MAPPO`, which uses global

Table 9: **Win rate of N-agent ad hoc teamwork.** We provide the average and 95% confidence interval of the win rate across six maps and six algorithms. This table includes the performance of self-play and XP.

Map	VDN	QMIX	IPPO	MAPPO	POAM	MA-JEPA
3m	85.5 ± 6.5	78.8 ± 7.7	73.8 ± 8.2	82.3 ± 6.8	81.8 ± 7.3	91.3 ± 5.2
8m	61.3 ± 9.3	55.7 ± 9.2	52.0 ± 9.2	65.2 ± 8.8	50.3 ± 9.8	65.5 ± 8.8
2s3z	95.3 ± 3.5	81.3 ± 5.3	87.7 ± 4.5	98.0 ± 1.8	86.3 ± 5.7	98.8 ± 1.2
5m_vs_6m	1.7 ± 1.0	4.0 ± 1.8	2.5 ± 2.3	3.8 ± 2.7	0.0 ± 0.0	4.5 ± 2.5
10m_vs_11m	18.7 ± 5.8	19.2 ± 6.0	12.7 ± 3.2	25.5 ± 5.8	0.5 ± 1.0	24.8 ± 5.0
27m_vs_30m	0.0 ± 0.0	0.0 ± 0.0	9.7 ± 2.8	18.8 ± 3.7	0.5 ± 0.5	22.5 ± 4.5

information in training. POAM is near zero on 5m_vs_6m and 10m_vs_11m against partners from other algorithms. QMIX has the highest self-play win rate on 5m_vs_6m but nearly fails in cross-play, a typical sign of conventions that do not transfer.

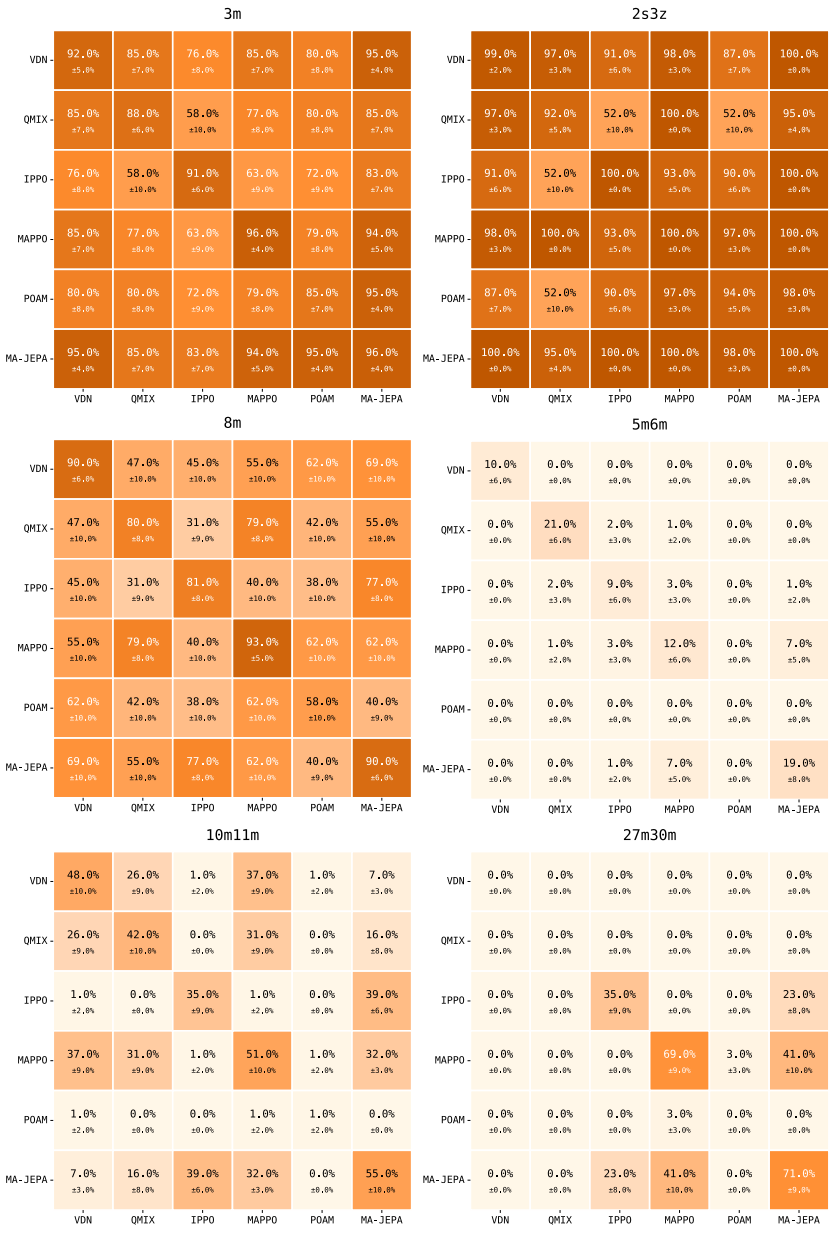

Figure 19: **N-agent ad hoc teamwork performance per map.**